Remote Control Manual

LeCroy

Innovators in Instrumentation

REMOTE CONTROL MANUAL

MODEL 9300 SERIES
DIGITAL OSCILLOSCOPES

Revision A – January 1993

LeCroy

Corporate Headquarters
700 Chestnut Ridge Road
Chestnut Ridge, NY 10977-6499
Tel: (914) 425-2000, TWX: 710-577-2832

European Headquarters
2, rue du Pré-de-la-Fontaine
P.O. Box 341
1217 Meyrin 1/Geneva, Switzerland
Tel.: (022) 719 21 11, Telex: 419 058

Copyright © January 1993, LeCroy. All rights reserved. Information in this publication supersedes all earlier versions. Specifications subject to change.

TABLE OF CONTENTS

1 General Information

Initial Inspection	1–1
Warranty	1–1
Product Assistance	1–1
Maintenance Agreements	1–1
Document Discrepancies	1–2
Service Procedure	1–2
Return Procedure	1–2

2 About Remote Control

GPIB Implementation Standard	2–1
Program Messages	2–1
Commands and Queries	2–2
Local and Remote State	2–2
Program Message Form	2–3
Command/Query Form	2–4
Response Message Form	2–7

3 GPIB Operation

GPIB Structure	3–1
Interface Capabilities	3–1
Addressing	3–2
GPIB Signals	3–2
IEEE 488.1 Standard Messages	3–3
Programming GPIB Transfers	3–5
Programming Service Requests	3–9
Instrument Polls	3–11
Driving a Hard–copy Device	3–15

4 RS–232–C Operation

Introduction	4–1
RS–232–C Pin Assignments	4–1
RS–232–C Configuration	4–2
Commands Simulating GPIB Commands	4–5

5 System Commands

Organization 5–1
Command Summary 5–1
Command Execution 5–3
Command Notation 5–4
General Instrument Reset 5–4

6 Waveform Structure

Introduction 6–1
Logical Data Blocks of a Waveform 6–1
Inspect? Command 6–2
Waveform? Command 6–4
Waveform Command 6–9
More Control of Waveform Queries 6–10
High–speed Waveform Transfer 6–10

7 Status Registers

Overview of Status and Service Request Reporting 7–1
Status Byte Register (STB) 7–3
Standard Event Status Register (ESR) 7–4
Standard Event Status Enable Register (ESE) 7–5
Service Request Enable Register (SRE) 7–5
Parallel Poll Enable Register (PRE) 7–5
Internal State Change Status Register (INR) 7–5
Internal State Change Enable Register (INE) 7–6
Command Error Status Register (CMR) 7–6
Device Dependent Error Status Register (DDR) 7–6
Execution Error Status Register (EXR) 7–6
User Request Status Register (URR) 7–6

Appendix A

Example 1: Use of the Interactive GPIB Program 'IBIC' A–1
Example 2: GPIB Program for IBM PC (High–level Function Calls) A–2
Example 3: GPIB Program for IBM PC (Low–level Function Calls) A–5

Appendix B

Waveform Template B–1

Digitized by the Internet Archive
in 2026

https://archive.org/details/manualsplus_11876

INITIAL INSPECTION

It is recommended that the shipment be thoroughly inspected immediately upon delivery to the purchaser. All material in the container should be checked against the enclosed Packing List. LeCroy cannot accept responsibility for shortages in comparison with the Packing List unless notified promptly. If the shipment is damaged in any way, please contact the Customer Service Department or local field office immediately.

WARRANTY

LeCroy warrants its oscilloscope products to operate within specifications under normal use for a period of two years from the date of shipment. Spares, replacement parts and repairs are warranted for 90 days. The instrument's firmware is thoroughly tested and thought to be functional, but is supplied "as is" with no warranty of any kind covering detailed performance. Products not manufactured by LeCroy are covered solely by the warranty of the original equipment manufacturer.

In exercising this warranty, LeCroy will repair or, at its option, replace any product returned to the Customer Service Department or an authorized service facility within the warranty period, provided that the warrantor's examination discloses that the product is defective due to workmanship or materials and that the defect has not been caused by misuse, neglect, accident or abnormal conditions or operation.

The purchaser is responsible for transportation and insurance charges for the return of products to the servicing facility. LeCroy will return all in–warranty products with transportation prepaid.

This warranty is in lieu of all other warranties, expressed or implied, including but not limited to any implied warranty of merchantability, fitness, or adequacy for any particular purpose or use. LeCroy shall not be liable for any special, incidental, or consequential damages, whether in contract or otherwise.

PRODUCT ASSISTANCE

Answers to questions concerning installation, calibration, and use of LeCroy equipment are available from the Customer Service Department, 700 Chestnut Ridge Road, Chestnut Ridge, New York 10977–6499, U.S.A., tel. (914)578–6061, and 2, rue du Pré–de–la–Fontaine, 1217 Meyrin 1, Geneva, Switzerland, tel. (41)22/719 21 11, or your local field engineering office.

MAINTENANCE AGREEMENTS

LeCroy offers a selection of customer support services. Maintenance agreements provide extended warranty and allow the customer to budget maintenance costs after the initial two year warranty has expired. Other services such as installation, training, enhancements and on–site repair are available through specific Supplemental Support Agreements.

DOCUMENTATION DISCREPANCIES

LeCroy is committed to providing state–of–the–art instrumentation and is continually refining and improving the performance of its products. While physical modifications can be implemented quite rapidly, the corrected documentation frequently requires more time to produce. Consequently, this manual may not agree in every detail with the accompanying product. There may be small discrepancies in the values of components for the purposes of pulse shape, timing, offset, etc., and, occasionally, minor logic changes. Where any such inconsistencies exist, please be assured that the unit is correct and incorporates the most up–to–date circuitry. In a similar way the firmware may undergo revision when the instrument is serviced. Should this be the case, manual updates will be made available as necessary.

SERVICE PROCEDURE

Products requiring maintenance should be returned to the Customer Service Department or authorized service facility. LeCroy will repair or replace any product under warranty at no charge. The customer is responsible for transportation charges to the factory. All in–warranty products will be returned to the customer with transportation prepaid.

For all LeCroy products in need of repair after the warranty period, the customer must provide a Purchase Order Number before repairs can be initiated. The customer will be billed for parts and labor for the repair, as well as for shipping.

RETURN PROCEDURE

To determine your nearest authorized service facility, contact the Customer Service Department or your field office. All products returned for repair should be identified by the model and serial numbers and include a description of the defect or failure, name and phone number of the user, and, in the case of products returned to the factory, a Return Authorization Number (RAN). The RAN may be obtained by contacting the Customer Service Department in New York, tel. (914)578–6061, in Geneva, tel. (41)22/719 21 11, or your nearest sales office.

Return shipments should be made prepaid. LeCroy will not accept C.O.D. or Collect Return Shipments. Air–freight is generally recommended. Wherever possible, the original shipping carton should be used. If a substitute carton is used, it should be rigid and be packed such that the product is surrounded with a minimum of four inches of excelsior or similar shock–absorbing material. In addressing the shipment, it is important that the Return Authorization Number be displayed on the outside of the container to ensure its prompt routing to the proper department within LeCroy.

Two modes of operation are available in the oscilloscope. The instrument may be operated either manually, by using the front–panel controls, or remotely by means of an external controller (which is usually a computer, but may be a simple terminal). This Remote Control Manual describes how to control the oscilloscope in the remote mode. For explanations on how to manually set front–panel controls, refer to the Operator's Manual.

The oscilloscope is remotely controlled via either the GPIB (General Purpose Interface Bus) or the RS–232–C communication ports. The instrument can be fully controlled in remote mode. The only actions which cannot be performed remotely are switching on the instrument or setting the remote address.

This chapter introduces the basic remote control concepts which are common to both RS–232–C and GPIB. It also presents a brief description of remote control messages.

Chapters 3 and 4 explain how to send program messages over the GPIB or the RS–232–C interfaces, respectively. Chapter 5 alphabetically lists all the remote control commands. Chapter 6 is a detailed description and tutorial of the transfer and format of waveforms, whereas Chapter 7 explains the use of status bytes for error reporting.

GPIB IMPLEMENTATION STANDARD

The remote commands conform to the GPIB IEEE 488.2 standard[1]. This standard may be seen as an extension of the IEEE 488.1 standard which dealt mainly with electrical and mechanical issues. The IEEE 488.2 recommendations have also been adopted for RS–232–C communications whenever applicable.

PROGRAM MESSAGES

To remotely control the oscilloscope the controller must send program messages which conform to precise format structures. The instrument will execute all program messages which are in the correct form and ignore those where errors are detected.

Warning or error messages are normally not reported by the instrument, unless the controller explicitly examines the relevant status register, or if the status enable registers have been set in such a way that the controller can be interrupted when an error occurs. The status registers are explained in Chapter 7.

1. ANSI/IEEE Std. 488.2–1987, "IEEE Standard Codes, Formats, Protocols, and Common Commands". The Institute of Electrical and Electronics Engineers Inc., 345 East 47th Street, New York, NY 10017, USA.

During the development of the control program it is possible to observe all remote control transactions, including error messages, on an external monitor connected to the RS–232–C port. Refer to the command "COMM_HELP" for further details.

COMMANDS AND QUERIES

Program messages consist of one or several commands or queries. A command directs the instrument to change its state, e.g. to change its time base or vertical sensitivity. A query asks the instrument about its state. Very often, the same mnemonic is used for a command and a query, the query being identified by a <?> after the last character.

For example, to change the time base to 2 msec/div, the controller should send the following command to the instrument

TIME_DIV 2 MS

To ask the instrument about its time base, this query should be sent

TIME_DIV?

A query causes the instrument to send a response message. The control program should read this message with a "read" instruction to the GPIB or RS–232–C interface of the controller. The response message to the query above might be

TIME_DIV 10 NS

The portion of the query preceding the question mark is repeated as part of the response message. If desired, this text may be suppressed with the command "COMM_HEADER".

Depending on the state of the instrument and the computation to be done, the controller may have to wait up to several seconds for a response. Command interpretation does not have priority over other oscilloscope activities. It is therefore judicious to set the controller IO timeout conditions to 3 or more seconds. In addition, it must be remembered that an incorrect query message will not generate a response message.

LOCAL AND REMOTE STATE

As a rule, remote commands are only executed by the instrument when it is in the REMOTE state, whereas queries are always executed. A few commands which don't affect the state of the front panel are also executed in LOCAL (refer to the beginning of Chapter 5 for a list of these commands). When the instrument is in REMOTE, all front–panel controls are disabled, except the menu buttons which can force a return to LOCAL mode and be used to communicate with the remote computer. For an explanation on how to set the instrument to LOCAL, REMOTE or LOCAL LOCKOUT, refer to Chapter 3 for GPIB and to Chapter 4 for RS–232–C.

PROGRAM MESSAGE FORM

An instrument is remotely controlled with program messages which consist of one or several commands or queries, separated by semicolons <;> and ended by a terminator:

<command/query>;.....;<command/query> <terminator>

Upper and/or lower case characters can be used for program messages.

The instrument does not decode an incoming program message before a terminator has been received (exception: if the program message is longer than the 256 byte input buffer of the instrument, the oscilloscope starts analyzing the message when the buffer is full). The commands or queries are executed in the order in which they are transmitted.

In GPIB mode, the following are valid terminators:

<NL>	New–line character (i.e. the ASCII new–line character, whose decimal value is 10).
<NL> <EOI>	New–line character with a simultaneous <EOI> signal.
<EOI>	<EOI> signal together with the last character of the program message.

Note: The <EOI> signal is a dedicated GPIB interface line which can be set with a special call to the GPIB interface driver. Refer to the GPIB interface manufacturer's manual and support programs.

The <NL> <EOI> terminator is always used in response messages sent by the instrument to the controller.

In RS–232–C, the terminator may be defined by the user with the command "COMM_RS232". The default value is <CR>, i.e. the ASCII carriage return character, the decimal value of which is 13.

Examples

GRID DUAL

This program message consists of a single command which instructs the instrument to display a dual grid. The terminator is not shown since usually it is automatically added by the interface driver routine which writes to the GPIB (or RS–232).

DZOM ON; DISPLAY OFF; DATE?

This program message consists of two commands, followed by a query. They instruct the instrument to turn on the multizoom mode, turn off the display, and then ask for the current date. Again, the terminator is not shown.

COMMAND/QUERY FORM

The general form of a command or a query consists of a command header <header> which is optionally followed by one or several parameters <data> separated by commas:

<header>[?] <data>,...,<data>

The notation [?] shows that the question mark is optional (turning the command into a query). The detailed listing of all commands in Chapter 5 indicates which commands may also be queries. There is a space between the header and the first parameter. There are commas between parameters.

Example

DATE 15,JAN,1992,13,21,16

This command instructs the oscilloscope to set its date and time to 15 JAN 1992, 13:21:16. The command header "DATE" indicates the action, the 6 data values specify it in detail.

Header

The header is the mnemonic form of the operation to be performed by the oscilloscope. All command mnemonics are listed in alphabetic order in Chapter 5.

The majority of the command/query headers have a long form for optimum legibility and a short form for better transfer and decoding speed. The two forms are fully equivalent and can be used interchangeably. For example, the following two commands for switching to the automatic trigger mode are fully equivalent:

TRIG_MODE AUTO and TRMD AUTO

Some command/query mnemonics are imposed by the IEEE 488.2 standard. They are standardized so that different instruments present the same programming interface for similar functions. All these mnemonics begin with an asterisk <*>, e.g. the command "*RST" is the IEEE 488.2 imposed mnemonic for resetting the instrument, whereas "*TST?" instructs the instrument to perform an internal self–test and to report the outcome.

Header path

Some commands or queries apply to a sub–section of the oscilloscope, e.g. a single input channel or a trace on the display. In such cases, the header must be preceded by a path name that indicates the channel or trace to which the command applies. The header path normally consists of a 2–letter path name followed by a colon <:> which immediately precedes the command header.

Usually one of the waveform traces can be specified in the header path (refer to the individual commands listed in Chapter 5 for details on which values apply to a given command header):

C1, C2	Channels 1 and 2
C3, C4	Channels 3 and 4 (in 4–channel instruments)
M1, M2, M3, M4	Memories 1, 2, 3, 4
TA, TB, TC, TD	Traces A, B, C and D
EX, EX10	External trigger
LINE	LINE source for trigger

Example

C1:OFST –300 MV Set the offset of Channel 1 to –300 mV

Header paths need only be specified once. Subsequent commands whose header destination is not indicated are assumed to refer to the last defined path. For example, the following commands are identical:

C2:VDIV?; C2:OFST?	What is the vertical sensitivity and the offset of channel 2?
C2:VDIV?; OFST?	Same as above, without repeating the path.

Data

Whenever a command/query uses additional data values, they are expressed in terms of ASCII characters. There is a single exception: the transfer of waveforms with the command/query "WAVEFORM", where the waveform may be expressed as a sequence of binary data values. Refer to Chapter 6 for a detailed explanation of the format of waveforms.

ASCII data can have the form of character, numeric, string or block data.

Character data

These are simple words or abbreviations for the indication of a specific action.

DUAL_ZOOM ON	The data value "ON" indicates that the dual–zoom mode should be turned on, rather than off.

In some commands, where as many as a dozen different parameters can be specified, or where not all parameters apply at the same time, the format requires pairs of data values. The first one names the parameter to be modified and the second gives its value. Only those parameter pairs to be changed need to be indicated.

HARDCOPY_SETUP DEV,EPSON,PORT,GPIB

Two pairs of parameters are specified. The first specifies the device as the EPSON printer (or compatible) and the second indicates the GPIB port. While the command "HARDCOPY_SETUP" allows many more parameters, they are either not relevant for printers or they are left unchanged.

Numeric Data

The numeric data type is used to enter quantitative information. Numbers can be entered as integers, as fractions or in exponential representation.

TA:VPOS –5	Move the displayed trace of Trace A downwards by 5 divisions.
C2:OFST 3.56	Set the DC offset of Channel 2 to 3.56 V.
TDIV 5.0E–6	Adjust the time base to 5 µsec/div.

Note: Numeric values may be followed by multipliers and units, modifying the value of the numerical expression. The following mnemonics are recognized:

EX	1E18	Exa–	PE	1E15	Peta–
T	1E12	Tera–	G	1E9	Giga–
MA	1E6	Mega–	K	1E3	kilo–
M	1E–3	milli–	U	1E–6	micro–
N	1E–9	nano–	PI	1E–12	pico–
F	1E–15	femto–	A	1E–18	atto–

For example, there are many ways of setting the time base of the instrument to 5 msec/div:

TDIV 5E–6	Exponential notation, without any suffix.
TDIV 5 US	Suffix multiplier "U" for 1E–6, with the (optional) suffix "S" for seconds.
TDIV 5000 NS	
TDIV 5000E–3 US	

String Data

This data type enables the transfer of a (long) string of characters as a single parameter. String data are formed by simply enclosing any sequence of ASCII characters between simple or double quotes.

MESSAGE 'Connect probe to point J3'

The instrument displays this message in the Message field above the grid.

Block Data

These are binary data values coded in hexadecimal ASCII, i.e. 4–bit nibbles are translated into the digits 0,...9, A,...F and transmitted as ASCII characters. They are only used for the transfer of waveforms (command "WAVEFORM") and of the instrument configuration (command "PANEL_SETUP")

RESPONSE MESSAGE FORM

The instrument sends a response message to the controller, as an answer to a query. The format of such messages is the same as that of program messages, i.e. individual responses in the format of commands, separated by semicolons <;> and ended by a terminator. They can be sent back to the instrument in the form in which they are received, and will be accepted as valid commands. In GPIB response messages, the <NL> <EOI> terminator is always used.

For example, if the controller sends the program message:

TIME_DIV?;TRIG_MODE NORM;C1:COUPLING? (terminator not shown)

the instrument might respond as follows:

TIME_DIV 50 NS;C1:COUPLING D50 (terminator not shown)

The response message only refers to the queries, i.e. "TRIG_MODE" is left out. If this response is sent back to the instrument, it is a valid program message for setting its time base to 50 nsec/div and the input coupling of Channel 1 to 50 Ω.

Whenever a response is expected from the instrument, the control program must instruct the GPIB or RS–232–C interface to read from the instrument. If the controller sends another program message without reading the response to the previous one, the response message in the output buffer of the instrument is discarded.

The instrument uses somewhat stricter rules for response messages than for the acceptance of program messages. Whereas the controller may send program messages in upper or lower case characters, response messages are always returned in upper case. Program messages may contain extraneous spaces or tabs (white space), response messages do not. Whereas program messages may contain a mixture of short and long command/query headers, response messages always use short headers as a default. However, the instrument can be forced with the command "COMM_HEADER" to use long headers or no headers at all. If the response header is omitted, the response transfer time is minimized, but such a response could not be sent back to the instrument again. In this case suffix units are also suppressed in the response.

If the trigger slope of Channel 1 is set to negative, the query "C1:TRSL?" could yield the following responses:

C1:TRIG_SLOPE NEG	header format: long
C1:TRSL NEG	header format: short
NEG	header format: off

Waveforms which are obtained from the instrument using the query "WAVEFORM?" constitute a special kind of response message. Their exact format can be controlled with the commands "COMM_FORMAT" and "COMM_ORDER", as explained in Chapter 6.

GPIB OPERATION

This section describes how to remotely control the oscilloscope via the GPIB. Topics discussed include interface capabilities, addressing, standard bus commands, and polling schemes.

GPIB STRUCTURE

The GPIB is like an ordinary computer bus, except that it interconnects independent devices via a cable bus whereas a computer has its circuit cards interconnected via a backplane bus. The GPIB carries program messages and interface messages:

- Program messages, often called device–dependent messages, contain programming instructions, measurement results, instrument status and waveform data. Their general form is described in chapter 2.
- Interface messages manage the bus itself. They perform functions such as initializing the bus, addressing and unaddressing devices and setting remote and local modes.

Devices on the GPIB can be listeners, talkers, and/or controllers. A talker sends program messages to one or more listeners. A controller manages the flow of information on the bus by sending interface messages to the devices.

The oscilloscope can be a talker or a listener, but not a controller. The host computer, however, must be able to act as a listener, talker and controller. For details on how the controller configures the GPIB for specific functions, refer to the GPIB interface manufacturer's manual.

INTERFACE CAPABILITIES

The interface capabilities of the oscilloscope include the following IEEE 488.1 definitions:

AH1	Complete Acceptor Handshake
SH1	Complete Source Handshake
L4	Partial Listener Function
T5	Complete Talker Function
SR1	Complete Service Request Function
RL1	Complete Remote/Local Function
DC1	Complete Device Clear Function
DT1	Complete Device Trigger
PP1	Parallel Polling: remote configurability
C0	No Controller Functions
E2	Tri–state Drivers

ADDRESSING

Every device on the GPIB has an address. When the remote control port is set to "GPIB" (UTILITIES menu), the instrument can be controlled via GPIB. When the remote control port is set to "RS–232", the instrument will only execute "talk–only" operations over the GPIB, such as driving a printer. Setting the oscilloscope to "RS–232" in the UTILITIES menu also enables the instrument to be controlled via the RS–232–C port.

If the oscilloscope is addressed to talk, it will remain configured to talk until a universal untalk command (UNT), its own listen address (MLA), or another instrument's talk address is received.

Similarly, if the oscilloscope is addressed to listen, it will remain configured to listen until a universal unlisten command (UNL), or its own talker address (MTA) is received.

GPIB SIGNALS

The bus system consists of 16 signal lines and 8 ground or shield lines. The signal lines are divided into 3 groups:

- 8 data lines
- 3 handshake lines
- 5 interface management lines

Data Lines

The eight data lines, usually called DI01 through DI08, carry both program and interface messages. Most of the messages use the 7–bit ASCII code, in which case DI08 is unused.

Handshake Lines

These three lines control the transfer of message bytes between devices. The process is called a three–wire interlocked handshake and it guarantees that the message bytes on the data lines are sent and received without transmission error.

Interface Management Lines

The following five lines manage the flow of information across the interface.

ATN (ATteNtion): The controller drives the ATN line true when it uses the data lines to send interface messages such as talk and listen addresses or a device clear (DCL) message. When ATN is false, the bus is in the data mode for the transfer of program messages from talkers to listeners.

IFC (InterFace Clear): The controller sets the IFC line true to initialize the bus.

REN (Remote ENable): The controller uses this line to place devices in remote or local program mode.

SRQ (Service ReQuest): Any device can drive the SRQ line true to asynchronously request service from the controller. This is the equivalent of a single interrupt line on a computer bus.

EOI (End Or Identify): This line has two purposes. The talker uses it to mark the end of a message string. The controller uses it to tell devices to identify their response in a parallel poll (discussed later in this section).

I/O Buffers

The instrument has a 256–byte input buffer and a 256–byte output buffer. An incoming program message is not decoded before a message terminator has been received. However, if the input buffer becomes full (because the program message is longer than the buffer), the instrument starts analyzing the message. In this case data transmission is temporarily halted, and the controller may generate a timeout if the limit was set too low.

IEEE 488.1 STANDARD MESSAGES

The IEEE 488.1 standard specifies not only the mechanical and electrical aspects of the GPIB, but also the low–level transfer protocol, e.g. it defines how a controller addresses devices, turns them into talkers or listeners, resets them or puts them in the remote state. Such interface messages are executed with the interface management lines of the GPIB, usually with ATN true.

All of these messages (except GET) are executed immediately upon reception and not in chronological order with normal commands.

Note: In addition to the IEEE 488.1 interface message standards, the IEEE 488.2 standard specifies some standardized program messages, i.e. command headers. They are identified with a leading asterisk <> and are listed among the commands in chapter 5.*

The command list in chapter 5 does not contain any command for clearing the input/output buffers or for setting the instrument to the remote state. This is because such commands are already specified as IEEE 488.1 standard messages. Refer to the GPIB interface manual of the host controller as well as to its support programs which should contain special calls for the execution of these messages.

The following describes those IEEE 488.1 standard messages which go beyond mere reconfiguration of the bus and which have an effect on the operation of the instrument.

Device Clear

In response to a universal Device CLear (DCL) or a Selected Device Clear message (SDC), the oscilloscope clears the input/output buffers, aborts the interpretation of the current command (if any) and clears any pending commands. Status registers and status enable registers are not cleared. Although DCL has an immediate effect it can take several seconds to execute this command if the instrument is busy.

Group Execute Trigger

The Group Execute Trigger message (GET) causes the oscilloscope to arm the trigger system. It is functionally identical to the "*TRG" command.

Remote ENable

This interface message is executed when the controller holds the Remote ENable control line (REN) true and configures the instrument as a listener. All the front–panel controls are disabled except the right–hand menu buttons. The menu indications on the right–hand side of the screen no longer appear since menus cannot now be operated manually. Instead, the text REMOTE ENABLE appears at the top of the menu field to indicate that the instrument is set to the remote mode. Whenever the controller returns the REN line to false, all instruments on the bus return to LOCAL. Individual instruments can be returned to LOCAL with the Go To Local message (see below).

As a rule, remote commands are only executed when the instrument is in the remote state, whereas queries are always executed. Local front–panel control may be regained by pressing the LOCAL menu button, unless the instrument was placed in the Local LOckout (LLO) mode.

Local LOckout

The Local LOckout command (LLO) causes the LOCAL menu button to disappear. The LLO command can be sent in local or remote mode but only becomes effective once the instrument has been set to the remote mode.

Go To Local

The Go To Local message (GTL) causes the instrument to return to the local mode. All front–panel controls become active and the normal menus reappear. Thereafter, whenever the instrument is addressed as a listener it will be immediately set to the remote state again.

Note that a GTL message does not clear the local lockout if it was set. Thus, whenever the instrument returns to the remote state the local lockout mode would immediately be effective again.

A command string should not be immediately followed by a GTL message. Since GTL is executed at once, the instrument may already be returned to the local state before the commands in the input buffer are interpreted. Therefore, the instrument may refuse to execute them if they require the instrument to be in REMOTE. A safe way to ensure that all commands have been interpreted is to append a query (e.g. "*STB?") to the command string and to wait for the response before sending a GTL.

InterFace Clear

The InterFace Clear message (IFC) initializes the GPIB but has no effect on the operation of the oscilloscope.

PROGRAMMING GPIB TRANSFERS

To illustrate the GPIB programming concepts a number of examples written in BASICA are included in this section. It is assumed that the controller is IBM–PC compatible, running under DOS, and that it is equipped with a National Instruments[2] GPIB interface card. GPIB programming with other languages such as C or Pascal is quite similar.

If you use another computer or another GPIB interface, refer to the interface manual for installation procedures and subroutine calls similar to those described here.

Configuring the GPIB Hardware

Check that the GPIB interface is properly installed in the computer. If it is not, follow the installation instructions of the interface manufacturer. In the case of the National Instruments interface, it is possible to modify the base I/O address of the board, the DMA channel number and the interrupt line setting using switches and jumpers. In our program examples, they are assumed to be left in their default positions.

Connect the oscilloscope to the computer with a GPIB interface cable. Set the GPIB address to the required value. The program examples assume that it is set to 4.

2. National Instruments Corporation, 12109 Technology Boulevard, Austin, Texas 78727

Configuring the GPIB Driver Software

The host computer needs an interface driver which handles the transactions between the user's programs and the interface board. In the case of the National Instruments interface, the installation procedure :

- copies the GPIB handler GPIB.COM into the boot directory.
- modifies the DOS system configuration file CONFIG.SYS to declare the presence of the GPIB handler.
- creates a sub–directory GPIB–PC.
- installs in GPIB–PC a number of files and programs which are useful for testing and reconfiguring the system, and for writing user programs.

The following files in the sub–directory GPIB–PC are of particular use:

IBIC.EXE allows interactive control of the GPIB via functions entered at the keyboard. Use of this program is highly recommended to anyone who is not familiar with GPIB programming or with the oscilloscope's remote commands.

DECL.BAS is a declaration file that contains code to be included at the beginning of any BASICA application program. Simple application programs can be quickly written by appending the user's instructions to DECL.BAS and executing the complete file.

IBCONF.EXE is an interactive program which allows inspection or modification of the current settings of the GPIB handler. To run IBCONF.EXE, refer to the National Instruments user's manual.

In the program examples in this section, it is assumed that the National Instruments GPIB driver GPIB.COM is in its default state, i.e. that the user has not modified it with IBCONF.EXE. This means that the interface board can be referred to by the symbolic name 'GPIB0' and that devices on the GPIB bus with addresses between 1 and 16 can be called by the symbolic names 'DEV1' to 'DEV16'.

Note: If you have a National Instruments PC2 interface card rather than PC2A, you must run IBCONF to declare the presence of this card rather than the default PC2A.

Simple Transfers

For a large number of remote control operations it is sufficient to use just 3 different subroutines (IBFIND, IBRD and IBWRT) provided by National Instruments. The following complete program reads the time–base setting of the oscilloscope and displays it on the terminal:

```
1–99    <DECL.BAS>
100     DEV$="DEV4"
110     CALL IBFIND(DEV$,SCOPE%)
120     CMD$="TDIV?"
130     CALL IBWRT(SCOPE%,CMD$)
140     CALL IBRD(SCOPE%,RD$)
150     PRINT RD$
160     END
```

Explanation

Lines 1 – 99 are a copy of the file DECL.BAS supplied by National Instruments. The first 6 lines are required for the initialization of the GPIB handler. The other lines are declarations which may be useful for larger programs, but are not really required code. The sample program above only uses the strings CMD$ and RD$ which are declared in DECL.BAS as arrays of 255 characters.

Note: DECL.BAS requires access to the file BIB.M during the GPIB initialization. BIB.M is one of the files supplied by National Instruments, and it must exist in the directory currently in use.

Note: The first 2 lines of DECL.BAS each contain a string "XXXXX" which must be replaced by the number of bytes which determine the maximum workspace for BASICA (computed by subtracting the size of BIB.M from the space currently available in BASICA). For example, if the size of BIB.M is 1200 bytes and when BASICA is loaded it reports "60200 bytes free", you should replace "XXXXX" by the value 59000 or less.

Lines 100 and 110 open the device "DEV4" and associate with it the descriptor "SCOPE%". All I/O calls from now on will refer to "SCOPE%". The default configuration of the GPIB handler recognizes "DEV4" and associates with it a device with GPIB address 4.

Lines 120 and 130 prepare the command string TDIV? and transfer it to the instrument. The command instructs it to respond with the current setting of the time base.

Line 140 reads the response of the instrument and places it into the character string RD$.

Line 150 displays the response on the terminal.

When running this sample program, the oscilloscope will automatically be set to the remote state when IBWRT is executed, and will remain in that state. Pressing the LOCAL menu button will return the oscilloscope to local mode if the GPIB handler was modified to inhibit Local LOckout (LLO).

Here is a slightly modified version of the sample program which checks if any error occurred during GPIB operation:

```
1–99      <DECL.BAS>
100       DEV$="DEV4"
110       CALL IBFIND(DEV$,SCOPE%)
120       CMD$="TDIV?"
130       CALL IBWRT(SCOPE%,CMD$)
140       IF ISTA% < 0 THEN GOTO 200
150       CALL IBRD(SCOPE%,RD$)
160       IF ISTA% < 0 THEN GOTO 250
170       PRINT RD$
180       IBLOC(SCOPE%)
190       END
200       PRINT "WRITE ERROR = ";IBERR%
210       END
250       PRINT "READ  ERROR = ";IBERR%
260       END
```

The GPIB status word ISTA%, the GPIB error variable IBERR% and the count variable IBCNT% are defined by the GPIB handler and are updated with every GPIB function call. Refer to the National Instruments user's manual for details. The sample program above would report if the GPIB address of the instrument was set to a value other then 4. Line 180 resets the instrument to local with a call to the GPIB routine IBLOC.

Some Additional Driver Calls

IBLOC is used to execute the IEEE 488.1 standard message Go To Local (GTL), i.e. it returns the instrument to the local state. The programming example above shows its use.

IBCLR executes the IEEE 488.1 standard message Selected Device Clear (SDC).

IBRDF and **IBWRTF** allow data to be read from GPIB to a file and data to be written from a file to GPIB respectively. Transferring data directly to or from a storage device does not limit the size of the data block, but it may be slower than transferring to the computer memory.

IBRDI and **IBWRTI** allow data to be read from GPIB to an integer array and data to be written from an integer array to GPIB. Since the integer array allows storage of up to 64 kilobytes (in BASIC), IBRDI and IBWRTI should be used for the transfer of large data blocks to the computer memory, rather than IBRD or IBWRT which are limited to 256 bytes by the BASIC string length. Note that IBRDI and IBWRTI only exist for BASIC, since the function calls IBRD and IBWRT for more modern programming languages, such as C, are much less limited in the data block size.

IBTMO can be used to change the time–out value during program execution. The default value of the GPIB driver is 10 seconds, e.g. if the instrument does not respond to a IBRD call, IBRD will return with an error after the specified time.

IBTRG executes the IEEE 488.1 standard message Group Execute Trigger (GET), which causes the oscilloscope to arm the trigger system.

National Instruments supply a number of additional function calls. In particular, it is possible to use the so–called board level calls which allow a very detailed control of the GPIB.

PROGRAMMING SERVICE REQUESTS

When an oscilloscope is used in a remote application, events often occur asynchronously, i.e. at times that are unpredictable for the host computer. The most common case is waiting for a trigger after the instrument has been armed. The controller must wait until the acquisition is finished before it can read the acquired waveform. The simplest way of checking if a certain event has occurred is by continuously or periodically reading the status bit associated with it until the required transition is detected. Continuous status bit polling is described in more detail in the sub–section "Instrument Polls". For a complete explanation of the status bytes refer to chapter 7.

A potentially more efficient way of detecting events occurring in the instrument is the use of the Service Request (SRQ). This GPIB interrupt line can be used to interrupt program execution in the controller. Therefore, the controller can execute other programs while waiting for the instrument. Unfortunately, not all interface manufacturers support the programming of interrupt service routines. In particular, National Instruments only supports the SRQ bit within the ISTA% status word. This requires the user to continuously or periodically check this word, either explicitly or with the function call IBWAIT. In the absence of real interrupt service routines the use of SRQ may not be very advantageous.

In the default state, after power–on, the Service ReQuest is disabled. The SRQ is enabled by setting the Service Request Enable register with the command "*SRE" and specifying which event should generate an SRQ. The oscilloscope will interrupt the controller as soon as the selected event(s) occur by asserting the SRQ interface line. If several devices are connected to the GPIB, the controller may have to identify which instrument caused the interrupt by serial polling the various devices.

*Note: The SRQ bit is latched until the controller reads the STatus Byte Register (STB). The action of reading the STB with the command "*STB?" clears the register contents except the MAV bit (bit 4) until a new event occurs. Service requesting may be disabled by clearing the SRE register ("*SRE 0").*

Example 1

To assert SRQ in response to the events "new signal acquired" or "return–to–local" (pressing the menu button LOCAL).

These events are tracked by the INR register which is reflected in the SRE register as the INB summary bit in position 0. Since the bit position 0 has the value 1, the command "*SRE 1" enables the generation of SRQ whenever the INB summary bit is set.

In addition, the events of the INR register which may be summarized in the INB bit must be specified. The event "new signal acquired" corresponds to INE bit 0 (value 1) while the event "return–to–local" is assigned to INE bit 2 (value 4). The total sum is 1+4=5. Thus the command "INE 5" is needed.

```
CMD$="INE 5;*SRE 1"

CALL IBWRT(SCOPE%,CMD$)
```

Example 2

To assert SRQ when soft key 4 is pressed.

The event "soft key 4 pressed" is tracked by the URR register. Since the URR register is not directly reflected in STB but only in the ESR register (URR, bit position 6), the ESE enable register must be set first with the command "*ESE 64" to allow the URQ setting to be reported in STB. An SRQ request will now be generated provided that the ESB summary bit (bit position 5) in the SRE enable register is set ("*SRE 32").

```
CMD$="*ESE 64;*SRE 32"

CALL IBWRT(SCOPE%,CMD$)
```

INSTRUMENT POLLS

State transitions occurring within the instrument can be remotely monitored by polling selected internal status registers. This sub–section discusses a number of polling methods which may be used to detect the occurrence of a given event.

1. Continuous poll
2. Serial poll
3. Parallel poll
4. *IST poll

To emphasize the differences between these methods, the same example will be presented in each case, i.e. determining if a new acquisition has taken place. By far the simplest poll is the continuous poll. The other methods only make sense if interrupt service routines (servicing the SRQ line) are supported or if multiple devices on GPIB must be monitored simultaneously.

Continuous Poll

In continuous polling a status register is continuously monitored until a transition is observed. This is the most straightforward method for detecting state changes but may be impracticable in some situations, especially in multiple device configurations.

In the following example, the event "new signal acquired" is observed by continuously polling the INternal state change Register (INR) until the corresponding bit (in this case bit 0, i.e. value 1) is non–zero to indicate that a new waveform has been acquired. Reading INR clears it at the same time so that there is no need for an additional clearing action after a non–zero value has been detected. The command "CHDR OFF" instructs the instrument to omit any command headers when responding to a query. This simplifies the decoding of the response. The instrument would therefore send "1" rather than "INR 1".

```
CMD$="CHDR OFF"
CALL IBWRT(SCOPE%,CMD$)
MASK% = 1 'New Signal Bit has value 1
LOOP% = 1
WHILE LOOP%
        CMD$="INR?"
        CALL IBWRT(SCOPE%,CMD$)
        CALL IBRD(SCOPE%,RD$)
        NEWSIG% = VAL(RD$) AND MASK%
        IF NEWSIG% = MASK% THEN LOOP% = 0
WEND
```

Serial Poll

Serial polling takes place once the SRQ interrupt line has been asserted. The controller examines which instrument has generated the interrupt by inspecting the SRQ bit in the STB register of each instrument. Because service request is based on an interrupt mechanism, serial polling offers a reasonable compromise in terms of servicing speed in multiple device configurations.

In the following example, the command "INE 1" enables the event "new signal acquired" to be reported in the INR to the INB bit of the status byte STB. The command "*SRE 1" enables the INB of the status byte to generate an SRQ whenever it is set. The function call IBWAIT instructs the computer to wait until one of three conditions occur: &H8000 in the mask (MASK%) corresponds to a GPIB error, &H4000 to a time–out error and &H0800 to the detection of RQS (ReQuest for Service generated by the SRQ bit).

Whenever IBWAIT detects RQS it automatically performs a serial poll to find out which instrument generated the interrupt. It will only exit if there was a time–out or if the instrument "SCOPE%" generated SRQ. The additional function call IBRSP fetches the value of the status byte which may be further interpreted. For this example to function properly the value of 'Disable Auto Serial Polling' must be set 'off' in the GPIB handler (use IBCONF.EXE to check).

```
CMD$="*CLS; INE 1; *SRE 1"
CALL IBWRT(SCOPE%,CMD$)
MASK% = &HC800
CALL IBWAIT(SCOPE%,MASK%)
IF (IBSTA% AND &HC000) <> 0 THEN PRINT "GPIB or Time–
    out Error" : STOP
CALL IBRSP(SCOPE%,SPR%)
PRINT "Status Byte = ", SPR%
```

*Note: After the serial poll is completed, the RQS bit in the STB status register is cleared. Note that the other STB register bits remain set until they are cleared by means of a "*CLS" command or the instrument is reset. If these bits are not cleared, they cannot generate another interrupt.*

Serial polling is only an advantage if there are several instruments that may need attention. Board–level function calls can deal simultaneously with several instruments attached to the same interface board. Refer to the National Instruments user's manual.

Parallel Poll

Parallel polling is only an advantage if there are several instruments that may need attention.

In parallel polling, the controller simultaneously reads the Individual STatus bit (IST) of all the instruments to determine which one needs service. Since parallel polling allows up to eight different instruments to be polled at the same time, parallel polling is the fastest way to identify state changes of instruments supporting this capability.

When a parallel poll is initiated, each instrument returns a status bit via one of the DIO data lines. Devices may respond either individually using a separate DIO line or collectively on a single data line. Data line assignments are made by the controller via a Parallel Poll Configure (PPC) sequence.

In the following example, the command "INE 1" enables the event "new signal acquired" in the INR to be reported to the INB bit of the status byte STB. The PaRallel poll Enable register (PRE) determines which events will be summarized in the IST status bit. The command "*PRE 1" enables the INB bit to set the IST bit whenever it is set. Once parallel polling has been established, the parallel poll status is examined until a change on data bus line DI02 takes place.

Stage 1: Enable the INE and PRE registers, configure the controller for parallel poll and instruct the oscilloscope to respond on data line 2 (DI02)

```
CMD1$="?_@$"
CALL IBCMD(BRD0%,CMD1$)
CMD$="INE 1;*PRE 1"
CALL IBWRT(BRD0%,CMD$)
CMD4$=CHR$(&H5)+CHR$(&H69)+"?"
CALL IBCMD(BRD0%,CMD4$)
```

Stage 2: Parallel poll the instrument until DI02 is set

```
LOOP% = 1
WHILE LOOP%
        CALL IBRPP(BRD0%,PPR%)
        IF (PPR% AND &H2) = 2 THEN LOOP% = 0
WEND
```

Stage 3: Disable parallel polling (hex 15) and clear the parallel poll register

```
CMD5$=CHR$(&H15)
CALL IBCMD(BRD0%,CMD5$)
CALL IBCMD(BRD0%,CMD1$)
CMD$="*PRE 0"
CALL IBWRT(BRD0%,CMD$)
```

Note 1: In the example above, board–level GPIB function calls are used. It is assumed that the controller (board) and oscilloscope (device) are respectively located at addresses 0 and 4. The listener and talker addresses for the controller and oscilloscope are:

Logic device	Listener address	Talker address
controller	32 (ASCII<space>)	64 (ASCII @)
oscilloscope	32+4=36 (ASCII $)	64+4=68 (ASCII D)

Note 2: The characters "?" and "_" appearing in the command strings stand for unlisten and untalk respectively. They are used to set the devices to a "known" state.

Note 3: To shorten the size of the program examples, device talking and listening initialization instructions have been grouped into character chains. They are:

```
CMD1$ = "?_@$" 'Unlisten, Untalk, PC talker, DSO listener
```

Note 4: The remote message code for executing a parallel response in binary form is 01101PPP where PPP specifies the data line. Since data line 2 is selected, the identification code is 001 which results in the code 01101001 (binary) or &H69 (hex). See Table 38 of the IEEE 488–1978 Standard for further details.

***IST Poll**

The state of the Individual STatus bit (IST) returned in parallel polling can also be read by sending the "*IST?" query. To enable this poll mode, the oscilloscope must be initialized as for parallel polling by writing into the PRE register. Since *IST polling emulates parallel polling, this method is applicable in all instances where parallel polling is not supported by the controller.

In the following example, the command "INE 1" enables the event "new signal acquired" in the INR to be reported to the INB bit of the status byte STB. The command "*PRE 1" enables the INB bit to set the IST bit whenever it is set. The command "CHDR OFF" suppresses the command header in the response of the instrument,

simplifying the interpretation. The status of the IST bit is then continuously monitored until it is set by the instrument.

```
CMD$="CHDR OFF; INE 1; *PRE 1"
CALL IBWRT(SCOPE%,CMD$)
LOOP% = 1
WHILE LOOP%
        CMD$="*IST?"
        CALL IBWRT(SCOPE%,CMD$)
        CALL IBRD(SCOPE%,RD$)
        IF VAL(RD$) = 1 THEN LOOP% = 0
WEND
```

DRIVING A HARDCOPY DEVICE

The oscilloscope can be interfaced to a wide range of hardcopy devices and be instructed to directly copy the screen contents onto these devices. The devices supported by the unit are listed with the command "HARDCOPY_SETUP" in chapter 5.

When the hardcopy device is connected to the GPIB two different configurations should be considered depending on whether or not a GPIB controller is available.

Printing without a GPIB Controller

When only the oscilloscope and the hardcopy device are connected to the GPIB, the oscilloscope must be configured as talker–only and the hardcopy device as listener–only to ensure proper data transfer. The oscilloscope can be configured as a talker–only by selecting the remote control port to RS–232 in the UTILITIES menu. The hardcopy device manufacturer usually specifies an address which forces the instrument into the listening mode.

- Select the remote control port to RS–232 (UTILITIES menu)
- Configure the "Hardcopy Setup" sub–menu in the UTILITIES menu specifying "GPIB" as hardcopy port
- Put the hardcopy device in listener–only mode
- Press the screen dump button on the front panel of the instrument

Printing with a GPIB Controller

If a controller is connected to the GPIB, data transfers must be supervised by the controller. Different schemes can be used to transfer the screen contents:

1. The controller reads the data into internal memory and then sends them to the printer. This alternative can be done with simple high–level GPIB function calls.
2. The oscilloscope sends data to both the controller and the printer.
3. The controller goes into a standby state. The oscilloscope becomes a talker and sends data directly to the printer.

1. Data read by controller and sent to printer

The controller stores the full set of printer instructions and sends them afterwards to the graphics device. This method is the most straightforward way of transferring screen contents but it requires a large amount of buffer storage.

```
CMD$ = "SCDP"
CALL IBWRT(SCOPE%,CMD$)
FILE$="PRINT.DAT"
CALL IBRDF(SCOPE%,FILE$)
CALL IBWRTF(PRINTER%,FILE$)
```

2. Oscilloscope sends data to controller and printer

The oscilloscope puts the printer instructions on to the bus. The data is directly put out and saved in scratch memory in the controller. The contents of the scratch file can be deleted later on.

Stage 1: Controller talker, oscilloscope listener. Issue the screen dump command

```
CMD1$="?_@$": CALL IBCMD(BRD0%,CMD1$)
CMD$="SCDP": CALL IBWRT(BRD0%,CMD$)
```

Stage 2: Oscilloscope talker, controller and printer listeners. Print data while storing data in scratch file SCRATCH.DAT

```
CMD2$="?_ D%": CALL IBCMD(BRD0%,CMD2$)
FILE$="SCRATCH.DAT": CALL IBRDF(BRD0%,FILE$)
```

3. Oscilloscope talks directly to printer

The controller goes into stand–by and resumes GPIB operations once the data have been printed, that is when an EOI is detected.

Stage 1: Controller talker, oscilloscope listener. Issue the screen dump command

```
CMD1$="?_@$": CALL IBCMD(BRD0%,CMD1$)
CMD$="SCDP": CALL IBWRT(BRD0%,CMD$)
```

Stage 2: Oscilloscope talker, printer listener. Put controller in stand–by

```
CMD2$="?_D%": CALL IBCMD(BRD0%,CMD2$)
V%=1: CALL IBGTS(BRD0%,V%)
```

Note 1: In schemes 2 and 3, board–level GPIB function calls are used. It is assumed that the controller (board), the oscilloscope and the printer are respectively located at addresses 0, 4 and 5. The listener and talker addresses for the controller, oscilloscope and printer are:

Logic device	Listener address	Talker address
controller	32 (ASCII<space>)	64 (ASCII @)
oscilloscope	32+4=36 (ASCII $)	64+4=68 (ASCII D)
hardcopy dev.	32+5=37 (ASCII %)	64+5=69 (ASCII E)

Note 2: The characters "?" and "_" appearing in the command strings stand for unlisten and untalk respectively. They are used to set the devices to a "known" state.

Note 3: To shorten the size of the program examples, device talking and listening initialization instructions have been grouped into character chains. They are:

```
CMD1$ = "?_@$" 'Unlisten, Untalk, PC talker, DSO listener
CMD2$ = "?_ D" 'Unlisten, Untalk, PC listener, DSO talker
```

INTRODUCTION

LeCroy oscilloscopes may be remotely controlled using a host, either a terminal or a computer, via the RS–232–C port. For this purpose the oscilloscope must be set to RS–232 control with the GPIB/RS232 submenu of the UTILITIES menu.

All the commands described in chapter 5 are supported but waveform transfer is only possible in HEX mode. The default value for COMM_FORMAT is set appropriately. The syntax of the response to WF? is identical to the GPIB case.

In this section some special RS–232–C commands are defined either for configuring the oscilloscope, or simulating GPIB 488.1 messages such as setting the oscilloscope into remote or local modes.

Notation

Throughout this section, characters which cannot be printed in ASCII will be represented by their mnemonics.

Example

<LF>	ASCII line feed character whose decimal value is 10
<BS>	ASCII backspace character whose decimal value is 8
CTRL_U	means that the control key and the U key are pressed simultaneously.

RS–232–C PIN ASSIGNMENTS

The remote RS–232–C pin assignments (indicated on the rear panel) are as follows:

DB9 Pin #		Description
3	T × D	Transmitted data (from the oscilloscope).
2	R × D	Received data (to the oscilloscope).
7	RTS	Request to send (from the oscilloscope). If the software Xon/Xoff handshake is selected it is always TRUE. Otherwise (hardware handshake) it is TRUE when the oscilloscope is able to receive characters and FALSE when the oscilloscope is unable to receive characters.
8	CTS	Clear to send (to the oscilloscope). When true, the oscilloscope can transmit; when false, transmission stops. It is used for the oscilloscope output hardware handshake.
4	DTR	Data terminal ready (from oscilloscope). Always TRUE.
—	GND	Protective Ground.
5	SIG GND	Signal Ground.

RS–232–C CONFIGURATION

The RS–232–C port is configured in full duplex. This means that the two sides (i.e. the controller and the oscilloscope) can both send and receive messages at the same time. However, when the oscilloscope receives a new command, it stops outputting.

Transmission of long messages to the oscilloscope should be done while the oscilloscope is in a triggered mode with no acquisition in progress. This is especially important when sending waveforms or front–panel setups into the oscilloscope.

The behavior of the RS–232–C port may be set according to the user's needs. For this purpose, in addition to the basic setup on the front–panel menu there are "immediate commands" as well as a special command "COMM_RS232". Immediate commands consist of the ASCII ESCape character <ESC> (whose decimal value is 27), followed by another character. Such commands are interpreted as soon as the second character has been received.

*Note: The RS–232–C baud rate, parity, character length and number of stop bits are among the parameters that are saved or recalled by the front–panel "SAVE" or "RECALL" button, or by the remote commands "*SAV", "*RCL" or "PANEL_SETUP". When recalling, care must be taken to ensure that these parameters are set at the same value as the actual ones. Otherwise, the host may no longer be able to communicate with the oscilloscope and a manual reconfiguration would be necessary.*

Echo of Received Characters

The serial port may echo the received characters. Echo is useful if the oscilloscope is attached to a terminal. Echoing can be turned on or off by sending the two character sequence <ESC>] or <ESC>[respectively. Echoing is on by default.

Note: The host must not echo characters received from the oscilloscope.

Handshake Control

When the oscilloscope input buffer becomes almost full, the instrument sends a handshake signal to the host telling it to stop transmitting. When this buffer has enough room to receive more characters another handshake signal will be sent. The handshake signals are either the CTRL–S (or <XOFF>) and CTRL–Q (<XON>) characters or a signal level on the RTS line. This is selected by sending the two–character sequence <ESC>) for XON/XOFF handshake – this is the default – or <ESC>(for RTS handshake.

The flow of characters coming from the oscilloscope may be controlled either by a signal level on the CTS line or by the <XON>/<XOFF> pair of characters.

Editing Features

When the oscilloscope is directly connected to a terminal, the following features will facilitate the correction of typing errors:

<BS> or <DELETE>	Delete the last character.
CTRL_U	Delete the last line.

Message Terminators

"Message terminators" are markers that indicate to the receiver that a message has been completed.

On input to the oscilloscope, the Program Message Terminator is one character which can be selected by the user. A good choice would be a character that is never used for anything else. The character is chosen using the command COMM_RS232 and the keyword EI. The default Program Message Terminator is the ASCII character <CR>, whose decimal value is 13.

The oscilloscope appends a Response Message Terminator to the end of each of its responses. It is a string, like a computer prompt, chosen by the user. This string must not be empty. The default Response Message Terminator is "\n\r" which means <LF><CR>.

Examples

(1) COMM_RS232 EI,3

This command informs the oscilloscope that each message it receives will be terminated with the ASCII character <ETX> which corresponds to 3 in decimal.

(2) COMM_RS232 EO,"\r\nEND\r\n"

This command indicates to the oscilloscope that it must append the string "\r\nEND\r\n" to each response.

After these settings, a host command will look like:

```
TDIV?<ETX>
```

The oscilloscope responds:

```
TDIV 1. S
END
```

Note: Having sent a COMM_RS232 command, the host must wait for the oscilloscope to change its behavior before sending a command in the new mode. A safe way to do this is to include a query on the line which contains the COMM_RS232 command and wait until the response is received. For example,

*COMM_RS232 EI,3; *STB?*

SRQ Message

Each time the Master Summary Status (MSS) bit of the STatus Byte (STB) is set, the SRQ message (a string of characters) is sent to the host to indicate that the oscilloscope requests service. The RS–232–C SRQ message has the same meaning as the GPIB SRQ message. If the string is empty, no message will be sent. This is the default setting. Note that no response message terminator is added at the end of the SRQ message.

Example

COMM_RS232 SRQ,"\r\n\nSRQ\r\n\a"

When the MSS bit is set, the oscilloscope will send

a <CR> followed by 2 <LF>s
SRQ
a <CR> followed by 1 <LF>

and the buzzer will sound.

Long Line Splitting

Line splitting is a feature provided for hosts that cannot accept lines with more than a certain number of characters. The oscilloscope may be configured to split responses into many lines. This feature is very useful for waveform or front–panel setup transfers although it is applicable to all response messages. Two parameters control this feature:

Line Separator:	Off	messages will not be split into lines
	<CR>,<LF> or <CR><LF>	possible line terminators.
Line length:	the maximum number of characters in a line.	

Example

COMM_RS232 LS,LF,LL,40

The line separator is the ASCII character <LF>, the line is a maximum of 40 characters long (excluding the line separator).

If the oscilloscope receives the command PNSU?, it may answer:

PNSU #9000001496
AAAA5555000655AA4030005800190000000000001
00000000000000000000000000C1B0100580000
00000000000000000000000000000000000000000
...

Remarks

Long commands sent to the oscilloscope may not be split into lines. If a command sent to the oscilloscope is the response to a previous query, the line split characters (<LF> and/or <CR>) must be removed.

This also applies to line split characters inside strings sent to the oscilloscope.

However, hex–ASCII data sent to the oscilloscope may contain line split characters. If you wish to use line splitting, ensure that neither the input message terminator characters nor the line split characters occur in the data.

COMMANDS SIMULATING GPIB COMMANDS

<ESC>C or **<ESC>c**
Device clear command

This command clears the input and output buffers. It has the same meaning as the GPIB DCL or SDC interface messages.

<ESC>R or **<ESC>r**
Set to remote command (REN)

This command puts the oscilloscope into the remote mode. Its function is the same as GPIB asserting the REN line and setting the oscilloscope to listener.

<ESC>L or **<ESC>l**
Set to local command

This command puts the oscilloscope into local mode. It clears local lockout. It has the same function as GPIB setting the REN line to false.

<ESC>F or **<ESC>f**
Set to local lockout command

This command disables the front–panel "LOCAL" button either immediately if the oscilloscope is already in the remote mode or later when the oscilloscope is next set to remote control. This disabling of the menu "LOCAL" button is called "Local Lockout" and can only be cancelled with the <ESC>L command. <ESC>F has the same meaning as the GPIB LLO interface message.

<ESC>T or **<ESC>t**
Trigger command (GET)

This command rearms the oscilloscope while it is in the "STOP" mode (valid only while the oscilloscope is in the remote mode). It has the same meaning as the "*TRG" command, and also the same meaning as the GPIB GET interface message.

SYSTEM COMMANDS

ORGANIZATION

This section of the manual lists all commands and queries recognized by the oscilloscope. For easy reference the listings are arranged in alphabetical order. Each command starts on a new page and the name (header) of the command is given in both the long and short forms. Below each name (header) it is indicated whether it denotes a command only, a command as well as a query, or a query only. For those headers that may be used to command an action, for example to modify a setup parameter, or to obtain some information such as the current value of a setup parameter, the query form is derived by appending a question mark (?) immediately to the header without intervening spaces.

The description of each command starts with a short explanation of the function performed by it, followed by a presentation of the formal syntax. In the formal syntax the header appears in mixed mode characters with the characters used to construct the short form shown in upper case.

Where applicable, the syntax of the query form is given along with the format of the response the oscilloscope will produce.

For most commands the description terminates with a short example illustrating a typical use of the command. The GPIB examples assume that the controller is equipped with a National Instruments interface board, and they show calls to the National Instruments interface subroutines in BASIC. The device name of the oscilloscope has been defined as "SCOPE%".

COMMAND SUMMARY

The following is an overview of the commands grouped according to their functionality.

Acquisition

To control the acquisition of waveforms:

ARM_ACQUISITION, AUTO_SETUP, BANDWIDTH_LIMIT, INTERLEAVED, MEMORY_SIZE, SEGMENTS, SEQUENCE, STOP, *TRG, WAIT

To select vertical input parameters to capture waveforms:

ATTENUATION, COUPLING, OFFSET, VOLT_DIV

To select time–base parameters to capture waveforms:

SAMPLE_CLOCK, TIME_DIV, TRIG_DELAY

To select trigger conditions to capture waveforms:

TRIG_COUPLING, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

Communication

To set communication characteristics:

COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER, COMM_RS232

Cursor	**To perform measurements:** CURSOR_MEASURE, CURSOR_SET, CURSOR_VALUE?, PARAMETER_CLR, PARAMETER_DELETE, PARAMETER_STATISTICS, PARAMETER_VALUE?, PASS_FAIL_CONDITION, PASS_FAIL_COUNTER, PASS_FAIL_DO, PASS_FAIL_MASK, XY_CURSOR_ORIGIN, XY_CURSOR_SET, XY_CURSOR_VALUE?
Display	**To display waveforms:** DISPLAY, DUAL_ZOOM, GRID, HOR_MAGNIFY, HOR_POSITION, INTENSITY, MULTI_ZOOM, PERSIST, PER_CURSOR _SET, PER_CURSOR_VALUE?, SELECT, TRACE, VERT_MAGNIFY, VERT_POSITION, XY_ASSIGN?, XY_DISPLAY **To display messages to a local user:** CALL_HOST, KEY, MESSAGE
Function	**To perform mathematical operations on waveforms:** CLEAR_SWEEPS, DEFINE, FUNCTION_RESET
Hard Copy	**To print the contents of the display screen:** HARDCOPY_SETUP, HARDCOPY_TRANSMIT, SCREEN_DUMP
Save/Recall Setup	**To preserve and restore front-panel settings:** PANEL_SETUP, *RCL, RECALL_PANEL, *RST, *SAV, STORE_PANEL
Status	**To obtain status information and set up service requests:** ALL_STATUS?, *CLS, CMR?, DDR?, *ESE, *ESR?, EXR?, INE, INR?, *IST?, *OPC, *PRE, *SRE, *STB?, URR?, *WAI
Waveform Transfer	**To preserve and restore waveforms:** INSPECT?, RECALL, STORE, STORE_SETUP, STORE_TEMPLATE, TEMPLATE?, WAVEFORM, WAVEFORM_SETUP, WAVEFORM_TEXT
Miscellaneous	**To control the calibration and test the instrument:** AUTO_CALIBRATE, *CAL?, CAL_OUTPUT, *TST? **To control the built-in buzzer:** (Hardware option CLBZ) BUZZER.

To control the real–time clock:

DATE

To delete a file from the memory card or floppy disk:

DELETE_FILE

To obtain a directory listing of the memory card or floppy disk:

DIRECTORY_LIST?

To format the memory card or floppy disk:

FORMAT_CARD, FORMAT_FLOPPY

To identify the instrument:

*IDN?, *OPT?

COMMAND EXECUTION

Before attempting to execute a command or query, the oscilloscope scans it to verify its correctness and that sufficient information is given to perform the requested action. To protect the local user from changes in the oscilloscope's behavior which are beyond his control, the remote user must set the oscilloscope to the remote state to execute commands that affect the operation of the instrument as an oscilloscope. If such a command is received while the oscilloscope is operating in the local state, an execution permission error is generated and the execution of the command is denied. Vice versa, the local user cannot interfere with the remote user because all front–panel controls are disabled while the oscilloscope is in the remote state.

Since interrogating the oscilloscope does not change its internal state, it may be queried at any time, independently of local or remote operation. There are only two exceptions to this rule: the queries *CAL? and *TST? both recalibrate the oscilloscope and are therefore executed in the remote state only.

Commands that only affect the remote behavior are executed independently of whether the oscilloscope is in the local or remote state. In this category are all commands that modify communication parameters (COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER, COMM_RS232), all commands affecting status information (*CLS, *ESE, INE, *OPC, *PRE, *SRE, *WAI), and the commands used to display messages on the screen to the local user (CALL_HOST, KEY, MESSAGE).

In the description of each command, only exceptions to the rule that a command is executed only in the remote state and a query is executed in both the local and remote states are mentioned.

COMMAND NOTATION

The following notation is used in the description of the individual commands :

< > Angular brackets enclose words that are used as placeholders. There are two types of placeholders: (1) the header path, (2) a data parameter of a command.

:= A colon followed by an equals sign separates a placeholder from the description of the type and range of values that may be used in a command instead of the placeholder.

{ } Braces enclose a list of choices from which one must be selected.

[ ] Square brackets enclose optional items.

... An ellipsis indicates that the items to the left and to the right of the ellipsis may be repeated zero or more times.

As an example, consider the syntax notation for the command to set the vertical input sensitivity:

<channel>:VOLT_DIV <v_gain>
<channel> := {C1, C2}
<v_gain> := 5.0 mV to 2.5 V

The first line shows the formal appearance of the command with <channel> denoting the placeholder for the header path, and <v_gain> denoting the placeholder for the data parameter specifying the desired vertical gain value. The second line indicates that either "C1" or "C2" must be chosen for the header path, and the third line explains that the actual vertical gain can be set to any value between 5 mV and 2.5 V.

GENERAL INSTRUMENT RESET

To reset the instrument, simultaneously press the "AUTO SETUP" button, the top menu–button, and the RETURN button.

STATUS

ALL_STATUS?, ALST?

Query

DESCRIPTION

The ALL_STATUS? query reads and clears the contents of all status registers: STB, ESR, INR, DDR, CMR, EXR, and URR except the MAV bit (bit 6) of the STB register. For an interpretation of the contents of each register, refer to the appropriate status register.

The ALL_STATUS? query is useful if a complete overview of the state of the instrument is required.

QUERY SYNTAX

ALl_STatus?

Response format

ALl_STatus STB,<value>,ESR,<value>,INR,<value>,
DDR,<value>, CMR<value>,EXR,<value>,URR,<value>

<value> := 0 to 65535

EXAMPLE (GPIB)

The following instruction reads the contents of all the status registers:

```
CMD$="ALST?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
ALST  STB,000000,ESR,000052,INR,000005,DDR,000000,
CMR,000004,EXR,000024,URR,000000
```

RELATED COMMANDS

*CLS, CMR?, DDR?, *ESR?, EXR?, *STB?, URR?

ACQUISITION

ARM_ACQUISITION, ARM

Command

DESCRIPTION

The ARM_ACQUISITION command enables the signal acquisition process by changing the acquisition state from "stopped" to "single".

COMMAND SYNTAX

ARM_acquisition

EXAMPLE

The following command enables signal acquisition:

```
CMD$ = "ARM": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STOP, *TRG, TRIG_MODE, WAIT

ACQUISITION

ATTENUATION, ATTN

Command/Query

DESCRIPTION

The ATTENUATION command selects the vertical attenuation factor of the probe. Values of 1, 10, 100, 1000 or 10000 may be specified.

The ATTENUATION? query returns the attenuation factor of the specified channel.

COMMAND SYNTAX

<channel>:ATTeNuation <attenuation>

<channel> := {C1, C2, C3*, C4*, EX, EX10}
<attenuation>:= {1, 10, 100, 1000, 10000}

QUERY SYNTAX

<channel>:ATTeNuation?

Response format

<channel>:ATTeNuation <attenuation>

EXAMPLE (GPIB)

The following command sets the attenuation factor of channel 1 to 100:

```
CMD$="C1:ATTN 100": CALL IBWRT(SCOPE%,CMD$)
```

* 4–channel oscilloscopes only

MISCELLANEOUS

AUTO_CALIBRATE, ACAL

Command/Query

DESCRIPTION

The AUTO_CALIBRATE command is used to enable or disable automatic calibration of the instrument. At power–up, auto–calibration is turned ON, i.e. all input channels are periodically calibrated for the current input amplifier and time–base settings.

The automatic calibration may be disabled by issuing the command ACAL OFF. Whenever it is convenient, a *CAL? query may be issued to fully calibrate the oscilloscope. When the oscilloscope is returned to local control, the periodic calibrations will be resumed.

The response to the AUTO_CALIBRATE? query indicates whether auto–calibration is enabled.

COMMAND SYNTAX

Auto_CALibrate <state>

<state> := {ON, OFF}

QUERY SYNTAX

Auto_CALibrate?

Response Format

Auto_CALibrate <state>

EXAMPLE (GPIB)

The following instruction disables auto–calibration:

CMD$="ACAL OFF": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

*CAL?

ACQUISITION

AUTO_SETUP, ASET

Command

DESCRIPTION

The AUTO_SETUP attempts to display the input signal(s) by adjusting the vertical, time–base and trigger parameters. Auto–setup operates only on the channels whose traces are currently turned on. If no traces are turned on, AUTO_SETUP operates on all channels and turns on all of the traces.

If signals are detected on several channels, the lowest numbered channel with a signal determines the selection of the time base and trigger source.

If only one input channel is turned on, the time base will be adjusted for that channel.

COMMAND SYNTAX

Auto_SETup

EXAMPLE

The following command instructs the oscilloscope to perform an auto–setup:

```
CMD$ = "ASET": CALL IBWRT(SCOPE%,CMD$)
```

ACQUISITION

BANDWIDTH_LIMIT, BWL

Command/Query

DESCRIPTION

The BANDWIDTH_LIMIT command enables or disables the bandwidth–limiting low–pass filter.

The response to the BANDWIDTH_LIMIT? query indicates if the bandwidth filter is on or off.

COMMAND SYNTAX

BandWidth_Limit <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

BandWidth_Limit?

Response Format

BandWidth_Limit <mode>

EXAMPLE

The following command turns the bandwidth filter on:

```
CMD$ = "BWL ON": CALL IBWRT(SCOPE%,CMD$)
```

MISCELLANEOUS

BUZZER, BUZZ

Command

DESCRIPTION

The BUZZER command controls the built–in piezo–electric buzzer. This may be useful to attract the attention of a local operator in an interactive working application. The buzzer may either be activated for short beeps (about 400 msec long in BEEP mode) or continuously for a certain time interval selected by the user by turning the buzzer ON or OFF. This command is only usable in oscilloscopes fitted with the CLBZ hard option.

Note: This command is always accepted (local and remote).

COMMAND SYNTAX

BUZZer := {BEEP, ON, OFF}

EXAMPLE (GPIB)

Sending the following code will cause the oscilloscope to sound two short tones.

```
CMD$ = "BUZZ BEEP; BUZZ BEEP";
CALL IBWRT(SCOPE%, CMD$)
```

MISCELLANEOUS

*CAL?

Query

DESCRIPTION

The *CAL? query performs a complete internal calibration. This calibration sequence is the same as that which occurs at power–up. At the end of the calibration, the response indicates how the calibration terminated. When the calibration is finished, the instrument returns to the state it was in prior to the query.

Hardware failures are identified by a unique binary code in the returned <status> number (see table below). A "0" response indicates that no failures occurred.

Note: This query is only accepted in remote mode.

QUERY SYNTAX

*CAL?

Response Format

*CAL <diagnostics>

<diagnostics> := 0 calibration successful

Bit	Bit Value	Description
0	1	CH1 failure
1	2	CH2 failure
2	4	CH3 failure
3	8	CH4 failure
4	16	TDC failure
5	32	Trigger circuit failure

Failures

EXAMPLE (GPIB)

The following instruction forces a self–calibration:

```
CMD$="*CAL?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure):
*CAL 0

RELATED COMMANDS

AUTO_CALIBRATE

MISCELLANEOUS

CAL_OUTPUT, COUT

Command/Query

DESCRIPTION

The CAL_OUTPUT command allows the user to set the type of signal put out at the CAL BNC connector.

COMMAND SYNTAX

Cal_OUTput <mode>,[<level>,[<rate>]]

<mode> := {OFF, CALSQ, CALPU, PF, TRIG, LEVEL, PULSE}
All oscilloscopes will allow CALSQ.
Oscilloscopes fitted with the CLBZ hardware option will allow OFF, CALPU, LEVEL, PULSE.
Only oscilloscopes fitted with the CKIO software option will allow PF and TRIG.

<level> := 0.0 to 1.00 V into 1 MΩ for oscilloscopes fitted with the CLBZ hardware option.

<freq> := 500 to 2 MHz for oscilloscopes fitted with the CLBZ hardware option.

QUERY SYNTAX

Cal_OUTput?

Response format

Cal_OUTput <mode>,<level>,[<rate>]

EXAMPLE (GPIB)

The following command sets the calibration signal to give a 0 – 0.2 volt pulse of 25 ns width at a 10 KHz rate.

```
CMD$="COUT CALPU,0.2 V,10 KHz":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO

DISPLAY

CALL_HOST, CHST

Command/Query

DESCRIPTION

The CALL_HOST command allows the user to manually generate a service request (SRQ). Once the CALL_HOST command has been received, the message "Call Host" will be displayed next to the lowest button in the menu field. Pressing this button while in the root menu causes the User Request status Register (URR) and the URQ bit of the Event Status Register to be set. This can generate a SRQ in local mode provided that the service request mechanism has been enabled.

The response to the CALL_HOST? query indicates whether Call Host is enabled (on) or disabled (off).

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

Call_HoST <state>

<state> := {ON, OFF}

QUERY SYNTAX

Call_HoST?

Response format

Call_HoST <state>

EXAMPLE (GPIB)

After executing the following code an SRQ request will be generated whenever the button is pressed. It is assumed that SRQ servicing has already been enabled.

```
CMD$="CHST ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

URR?

FUNCTION

CLEAR_SWEEPS, CLSW

Command

DESCRIPTION

The CLEAR_SWEEPS command restarts the cumulative processing functions: summed or continuous average, extrema, FFT power average, pulse parameter statistics, pass/fail counters, and persistence.

COMMAND SYNTAX

CLear SWeeps

EXAMPLE (GPIB)

The following example will restart the cumulative processing:

```
CMD$ = "CLSW": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, INR?

STATUS

*CLS
Command

DESCRIPTION

The *CLS command clears all the status data registers.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

*CLS

EXAMPLE (GPIB)

The following command causes all the status data registers to be cleared:

```
CMD$ = "*CLS": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

ALL_STATUS?, CMR?, DDR?, *ESR?, EXR?, *STB?, URR?

STATUS

CMR?

Query

DESCRIPTION

The CMR? query reads and clears the contents of the CoMmand error Register (CMR). The CMR register (see table below) specifies the last syntax error type detected by the instrument.

QUERY SYNTAX

CMR?

Response format

CMR <value>

<value> := 1 to 13

EXAMPLE (GPIB)

The following instruction reads the contents of the CMR register:

```
CMD$="CMR?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

CMR 0

RELATED COMMANDS

ALL_STATUS?, *CLS

Value	Description
1	Unrecognized command/query header
2	Illegal header path
3	Illegal number
4	Illegal number suffix
5	Unrecognized keyword
6	String error
7	GET embedded in another message
10	Arbitrary data block expected
11	Non–digit character in byte count field of arbitrary data block
12	EOI detected during definite length data block transfer
13	Extra bytes detected during definite length data block transfer

Command Error Status Register Structure (CMR)

COMMUNICATION

COMM_FORMAT, CFMT

Command/Query

DESCRIPTION

The COMM_FORMAT command selects the format which the oscilloscope will use to send waveform data. The available options allow (1) the block format, (2) the data type and (3) the encoding mode to be modified from the default settings.

Note: This command can be executed in both local and remote modes.

The COMM_FORMAT? query returns the currently selected waveform data format.

COMMAND SYNTAX

Comm_ForMaT <block_format>,<data_type>,<encoding>

<block_format> := {DEF9, IND0, OFF}
<data_type> := {BYTE, WORD}
<encoding> := {BIN, HEX}
(GPIB uses both encoding forms, RS–232–C always uses HEX)

Initial settings (i.e. after power on) are:

DEF9, WORD, BIN for GPIB
DEF9, WORD, HEX for RS–232–C

QUERY SYNTAX

Comm_ForMaT?

Response Format

Comm_ForMaT <block_format>,<data_type>,<encoding>

EXAMPLE (GPIB)

The following code redefines the transmission format of waveform data. The data will be transmitted as a block of indefinite length. Data will be coded in binary and represented as 8–bit integers.

```
CMD$="CFMT IND0,BYTE,BIN"
CALL IBWRT(SCOPE%,CMD$)
```

EXPLANATION

BLOCK FORMAT

DEF9: Uses the IEEE 488.2 definite length arbitrary block response data format. The digit 9 indicates that the byte count consists of 9 digits. The data block directly follows the byte count field.

For example, a data block consisting of 3 data bytes would be sent as:

WF DAT1,#9000000003<DAB><DAB><DAB>
where <DAB> represents an 8–bit binary data byte.

IND0: Uses the IEEE 488.2 indefinite length arbitrary block response data format.

A <NL^END> (new line with EOI) signifies that block transmission has ended.

The same data bytes as above would be sent as:

WF DAT1,#0<DAB><DAB><DAB><NL^END>

OFF: Same as IND0. In addition, the data block type identifier and the leading #0 of the indefinite length block will be suppressed. The data presented above would be sent as:

WF <DAB><DAB><DAB><NL^END>

Note: The format OFF does not conform to the IEEE 488.2 standard and is only provided for special applications where the absolute minimum of data transfer may be important.

DATA TYPE

BYTE: Transmits the waveform data as 8–bit signed integers (1 byte).

WORD: Transmits the waveform data as 16–bit signed integers (2 bytes).

Note: The data type BYTE transmits only the high order bits of the internal 16–bit representation. The precision contained in the low order bits is lost.

ENCODING

BIN: Binary encoding (GPIB only)

HEX: Hexadecimal encoding (bytes are converted to 2 hexadecimal ASCII digits (0, ... 9, A, ... F))

RELATED COMMANDS WAVEFORM?

COMMUNICATION

COMM_HEADER, CHDR

Command/Query

DESCRIPTION

The COMM_HEADER command controls the way the oscilloscope will format responses to queries. The instrument provides three response formats: (1) LONG format, i.e. responses start with the long form of the header word; (2) SHORT format, i.e. responses start with the short form of the header word; (3) OFF, i.e. headers are omitted from the response and suffix units in numbers are suppressed. Until the user requests otherwise, the SHORT response format is used.

This command does not affect the interpretation of messages sent to the oscilloscope. Headers may be sent in their long or short form regardless of the COMM_HEADER setting.

Querying the vertical sensitivity of Channel 1 may result in one of the following responses:

COMM_HEADER	RESPONSE
LONG	CHANNEL_1:VOLT_DIV 200E–3 V
SHORT	C1:VDIV 200E–3 V
OFF	200E–3

.Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

Comm_HeaDeR <mode>

<mode> := {SHORT, LONG, OFF}

Note: The default mode, i.e. the mode just after power on, is SHORT.

QUERY SYNTAX

Comm_HeaDeR?

Response Format

Comm_HeaDeR <mode>

EXAMPLE (GPIB)

The following code sets the response header format to short:

CMD$ = "CHDR SHORT"; CALL IBWRT(SCOPE%,CMD$)

COMMUNICATION

COMM_HELP, CHLP

Command/Query

DESCRIPTION

The COMM_HELP command enables the help diagnostics utility to assist remote program debugging. When turned on, this utility displays all message transactions occurring between the controller and the oscilloscope on a terminal, printer or similar recording device connected to the RS–232–C port. Errors detected by the instrument can be directly viewed.

Note: This command can be executed in both local and remote modes.

The COMM_HELP? query indicates if the diagnostics utility has been enabled.

COMMAND SYNTAX

Comm_HeLP <target>

<target> := {RS, OFF}

The initial <target>, (i.e. after power on) is OFF.

QUERY SYNTAX

Comm_HeLP?

Response Format

Comm_HeLP <target>

EXAMPLE (GPIB)

The following code turns on the remote control diagnostics utility:

CMD$="CHLP RS"; CALL IBWRT(SCOPE%,CMD$)

COMMUNICATION

COMM_ORDER, CORD

Command/Query

DESCRIPTION

The COMM_ORDER command controls the byte order of waveform data transfers. Waveform data may be sent with the most significant byte (MSB) or the least significant byte (LSB) in the first position. The default mode is the MSB first.

COMM_ORDER applies equally to the waveform's descriptor and time blocks. In the descriptor some values are 16 bits long ("word"), 32 bits long ("long " or "float"), or 64 bits long ("double"). In the time block all values are floating values, i.e. 32 bits long. When "COMM_ORDER HI" is selected the most significant byte is sent first. When "COMM_ORDER LO" is specified the least significant byte is sent first.

The COMM_ORDER? query returns the byte transmission order currently in use.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

Comm_ORDer <mode>

<mode> := {HI, LO}

Note: The initial mode, i.e. the mode after power on, is HI.

QUERY SYNTAX

Comm_ORDer?

Response Format

Comm_ORDer <mode>

EXAMPLE

The order of transmission of waveform data depends on the data type. The following table illustrates the different possibilities.

Type	CORD HI	CORD LO
Word	<MSB><LSB>	<LSB><MSB>
Long/float	<MSB><byte 2><byte 3><LSB>	<LSB><byte 3><byte 2><MSB>
Double	<MSB><byte 2> ... <byte 7><LSB>	<LSB><byte 7> ... <byte 2><MSB>

Waveform Data Transmission Order

RELATED COMMANDS

WAVEFORM?

COMMUNICATION

COMM_RS232, CORS

Command/Query

DESCRIPTION

The command COMM_RS232 sets the parameters of the RS–232–C port for remote control.

The COMM_RS232? query reports the settings of the parameters.

Note: This command is ONLY valid if the oscilloscope is being remotely controlled via the RS–232–C port.

The parameters are:

a. End Input character. When received by the oscilloscope, this character will be interpreted as the END–of–a–command message marker. The commands received will be parsed and executed.

b. End Output string. The oscilloscope will add this string at the end of a response message. When the host computer receives this string, it knows that the oscilloscope has completed its response.

c. Line Length. This parameter defines the maximum number of characters that will be sent to the host in a single line. Remaining characters of the response will be output in separate additional lines. This parameter is only applicable if a line separator has been selected.

d. Line Separator. This parameter is used to select the line splitting mechanism and to define the characters used to split the oscilloscope response messages into many lines. Possible line separators are: CR, LF, CRLF. A <CR>, a <LF> or a <CR> followed by a <LF> will be sent to the host computer after <line_length> characters.

e. SRQ string. This string is sent each time the oscilloscope wants to signal an SRQ to the host computer.

Note: Some parameters of this command require ASCII strings as actual arguments. In order to facilitate the embedding of non–printable characters into such strings, escape sequences may be used.

the back–slash character ('\') is used as an escape character. The following escape sequences are recognized:

"\a": Bell character,

"\b": Back space character,

"\e": Escape character,

"\n": Line feed character,

"\r": Carriage return character,

"\t": Horizontal tab character,

"\\": The back–slash character itself

"\ddd": ddd represents one to three decimal digit characters giving the code value of the corresponding ASCII character. This allows any ASCII code in the range 1 to 127 to be inserted.

Before using the string, the oscilloscope will replace the escape sequence by the corresponding ASCII character.

For example, the escape sequences "\r", "\13" and "\013" are all replaced by the single ASCII character <Carriage Return>.

Notation			
EI	End input character	EO	End output string
LL	Line length	LS	Line separator
SRQ	SRQ Service request		

COMMAND SYNTAX

COmm_RS232 EI,<ei_char>,EO,'<eo_string>',LL,<line_length>, LS,<Line_sep>,SRQ,'<srq_string>'

<ei_char> := 1 to 126 (default: 13 = <CR>)

<eo_string> := A non–empty ASCII string of up to 20 characters. (default: "\n\r")

<line_length> := 40 to 1024 (default: 256)

<line_sep> := {OFF, CR, LF, CRLF} (default: OFF)

<srq_string> := An ASCII string which may be empty. (default: empty string)

QUERY SYNTAX

COmm_RS232?

Response Format

COmm_RS232 EI,<ei_char>,EO,"<eo_string>",
LL,<line_length>,LS,<line_sep>,SRQ,"<srq_string>"

EXAMPLE

After executing the command:

COMM_RS232 EI,3,EO,"\R\NEND\R\N"

the oscilloscope will assume that it has received a complete message each time the <ETX> (decimal value 3) is detected. Response messages will be terminated by sending the character sequence "<CR><LF>END<CR><LF>".

ACQUISITION

COUPLING, CPL

Command/Query

DESCRIPTION

The COUPLING command selects the coupling mode of the specified input channel.

The COUPLING? query returns the coupling mode of the specified channel.

COMMAND SYNTAX

<channel>:CouPLing <coupling>

<channel> := {C1, C2, C3*, C4*}
<coupling> := {A1M, D1M, D50, GND}

QUERY SYNTAX

<channel>:CouPLing?

Response format

<channel>:CouPLing <coupling>

In the event of signal overload while in the DC 50 Ω coupling the oscilloscope will disconnect the input. In such a case:
<coupling> := OVL

EXAMPLE GPIB)

The following command sets the coupling of Channel 2 to 50 Ω DC:

```
CMD$="C2:CPL D50": CALL IBWRT(SCOPE%,CMD$)
```

* 4–channel oscilloscopes only

CURSOR

CURSOR_MEASURE, CRMS

Command/Query

DESCRIPTION

The CURSOR_MEASURE command specifies the type of cursor or parameter measurement to be displayed.

The CURSOR_MEASURE? query indicates which cursors or parameter measurements are currently displayed.

Notation	
CUST	custom parameters
FAIL	pass–fail: fail
HABS	horizontal absolute cursors
HPAR	standard time parameters
HREL	horizontal relative cursors
OFF	cursors and parameters off
PARAM	synonym for VPAR
PASS	pass–fail: pass
SHOW	custom parameters (old form)
STAT	parameter statistics
VABS	vertical absolute cursors
VPAR	standard voltage parameters
VREL	vertical relative cursors

Note: The PARAM mode is turned OFF when the XY or persistence modes are ON.

COMMAND SYNTAX

CuRsor_MeaSure <mode> [,STAT]

<mode> := {CUST, FAIL, HABS, HPAR, HREL, OFF, PARAM, PASS, SHOW, VABS, VPAR, VREL}

Note: The keyword STAT is optional with modes CUST, HPAR, and VPAR. If present, STAT turns parameter statistics on. Absence of STAT turns parameter statistics off.

QUERY SYNTAX

CuRsor_MeaSure?

Response Format

CuRsor_MeaSure <mode>

EXAMPLE (GPIB)

The following command switches on the vertical relative cursors:

```
CMD$="CRMS VREL": CALL IBWRT(SCOPE%,CMD$)
```

The following command determines which cursor is currently turned on:

```
CMDS$="CRMS?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Example of response message:

```
CRMS OFF
```

RELATED COMMANDS

CURSOR_SET, PARAMETER_VALUE?, PASS_FAIL_CONDITION, PASS_FAIL_DELETE, PASS_FAIL_CLEAR, PASS_FAIL_MASK, PARAMETER_STATISTICS

EXPLANATION

To turn off the cursors, parameter measurements or Pass/Fail tests, use:

```
CURSOR_MEASURE OFF
```

To turn on a cursor display, use one of the following four forms:

```
CURSOR_MEASURE HABS
CURSOR_MEASURE HREL
CURSOR_MEASURE VABS
CURSOR_MEASURE VREL
```

To turn on parameter measurements without statistics, use one of the following three forms:

```
CURSOR_MEASURE CUST
CURSOR_MEASURE HPAR
CURSOR_MEASURE VPAR
```

To turn on parameter statistics, add the keyword STAT to the above three forms.

To turn on Pass or Fail tests on parameter or mask tests, use :

```
CURSOR_MEASURE PASS
CURSOR_MEASURE FAIL
```

Use the command:

```
PASS_FAIL_CONDITION
```

to select parameters in the Custom mode, and to modify the test conditions in the Pass/Fail mode.

CURSOR

CURSOR_SET, CRST

Command/Query

DESCRIPTION

The CURSOR_SET command allows the user to position any one of the eight independent cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed on the screen.

When setting a cursor position, a trace must be specified, relative to which the cursor will be positioned.

The CURSOR_SET? query indicates the current position of the cursor(s). The values returned depend on the grid type selected.

Note: If the parameter display is turned on (or the pass/fail display or the extended parameters display), the parameters of the specified trace will be shown unless the newly chosen trace is not displayed or has been acquired in sequence mode (these conditions will produce an environment error, (see table on page 5–49). To only change the trace without repositioning the cursors, the CURSOR_SET command may be given with no argument, (e.g. TB:CRST).

Notation			
HABS	Horizontal absolute	HDIF	Horizontal difference
HREF	Horizontal reference	PDIF	Parameter difference
PREF	Parameter reference	VABS	Vertical absolute
VDIF	Vertical difference	VREF	Vertical reference

COMMAND SYNTAX

<trace>:CuRsor_SeT <cursor>,<position>[<cursor>,<position>, ...<cursor>,<position>]

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}
<cursor> := {HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF}
<position> := 0 to 10 DIV (horizontal)
–29.5 to 29.5 DIV (vertical)

Note 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first parameter specifies the cursor to be modified and the second one indicates its new value. Parameters may be grouped in any order and may be restricted to those items to be changed.

* 4–channel oscilloscopes only

QUERY SYNTAX

<trace>:CuRsor_SeT? [<cursor>,...<cursor>]

<cursor>:= {HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF, ALL}

Response Format

<trace>:CuRsor_SeT <cursor>,<position>[<cursor>,<position>, ...<cursor>,<position>]

If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.

EXAMPLE (GPIB)

The following command positions the VREF and VDIF cursors at +3 DIV and –7 DIV respectively, using Trace A as a reference:

```
CMD$="TA:CRST VREF,3DIV,VDIF,–7DIV"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR VALUE?, PARAMETER_VALUE?, PER_CURSOR_SET, XY_CURSOR_SET

CURSOR

CURSOR_VALUE?, CRVA?

Query

DESCRIPTION

The CURSOR_VALUE? query returns the values measured by the specified cursors for a given trace. (The PARAMETER_VALUE? query is used to obtain measured waveform parameter values.)

Notation			
HABS	Horizontal absolute	HREL	Horizontal relative
VABS	Vertical absolute	VREL	Vertical relative

QUERY SYNTAX

<trace>:CuRsor_VAlue? [<mode>,...<mode>]

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}
<mode>:= {HABS, HREL, VABS, VREL, ALL}

Response Format

<trace>:CuRsor_VAlue? <mode>[,<hor_value>],<ver_value>
[,...,<mode>[,<hor_value>],<ver_value>]

For horizontal cursors, both horizontal as well as vertical values are given, whereas for vertical cursors only vertical values are given.

Note: If <mode> is not specified or equals ALL, all the measured cursor values for the specified trace are returned. If the value of a cursor could not be determined in the current environment, the value UNDEF will be returned.

EXAMPLE (GPIB)

The following query reads the measured absolute horizontal value of the cross–hair cursor (HABS) on Channel 2:

```
CMD$="C2:CRVA? HABS": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:
C2:CRVA HABS,34.2E–6 S, 244 E–3 V

RELATED COMMANDS

CURSOR_SET, PARAMETER_VALUE?,
PER_CURSOR_VALUE?, XY_CURSOR VALUE?,

* 4–channel oscilloscopes only

MISCELLANEOUS

DATE

Command/Query

DESCRIPTION

The DATE command changes the date/time of the oscilloscope's internal real time clock.

The DATE? query returns the current date/time setting.

COMMAND SYNTAX

DATE <day>,<month>,<year>,<hour>,<minute>,<second>

<day> := 1 to 31
<month> := {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC}
<year> := 1990 to 2089
<hour> := 0 to 23
<minute> := 0 to 59
<second> := 0 to 59

Note: It is not always necessary to specify all the DATE parameters. Only the parameters up to and including the parameter to be changed need to be specified, i.e. to change the "year" setting specify day, month and year together with the required settings. The time settings will remain unchanged. To change the "second" setting all the DATE parameters must be specified with the required settings.

QUERY SYNTAX

DATE?

Response format

DATE <day>,<month>,<year>,<hour>,<minute>,<second>

EXAMPLE (GPIB)

This example will change the date to January 1, 1992 and the time to 1:21:16 p.m. (13:21:16 in 24–hour notation):

```
CMD$="DATE 1,JAN,1992,13,21,16"
CALL IBWRT(SCOPE%,CMD$)
```

STATUS

DDR?

Query

DESCRIPTION

The DDR? query reads and clears the contents of the Device Dependent or device specific error Register (DDR). In the case of a hardware failure, the DDR register specifies the origin of the failure. See table below for further details.

QUERY SYNTAX

DDR?

Response format

DDR <value>

<value> := 0 to 65535

EXAMPLE (GPIB)

The following instruction reads the contents of the DDR register:

```
CMD$="DDR?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:
DDR 0

RELATED COMMANDS

ALL_STATUS?, *CLS

Bit	Bit Value		Description
15...14		0	Reserved
13	8192	1	A timebase hardware failure is detected
12	4096	1	A trigger hardware failure is detected
11	2048	1	A Channel 4* hardware failure is detected
10	1024	1	A Channel 3* hardware failure is detected
9	512	1	A Channel 2 hardware failure is detected
8	256	1	A Channel 1 hardware failure is detected
7..4		0	Reserved
3	8	1	A Channel 4* overload condition is detected
2	4	1	A Channel 3* overload condition is detected
1	2	1	A Channel 2 overload condition is detected
0	1	1	A Channel 1 overload condition is detected

** 4–channel oscilloscopes only*

Device Specific Register Structure (DDR)

FUNCTION

DEFINE, DEF

Command/Query

Standard oscilloscopes

DESCRIPTION

The DEFINE command specifies the mathematical expression to be evaluated by a function.

Notation			
AVGS	average summed	EQN	equation
C1	Channel 1	C2	Channel 2
C3*	Channel 3	C4*	Channel 4
MAXPTS	maximum number of points		
SWEEPS	maximum number of sweeps		
ZOOMONLY	No Math		

COMMAND SYNTAX

<function>:DEFine EQN,'<equation>',MAXPTS,<max_points>,
SWEEPS,<max_sweeps>
<function> := {TA, TB, TC, TD}
<equation> := {AVGS(<source>),
<source> + <source>, + <source>,
<source> – <source>, – <source>,
<source> * <source>,
<source> / <source>,ZOOMONLY (<source>)}
<source> := {C1, C2, C3*, C4*}

<max_points> :=	L models	M models	Others
	50000	50000	10000

<max_sweeps> := 1 to 1000

Note 1: Parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be given in any order and may be restricted to the variables to be changed.

Note 2: The pair SWEEPS,<max_sweeps> applies only to the summed averaging (AVGS).

QUERY SYNTAX

<function>:DEFine?

Response format

<function>:DEFine EQN,'<equation>',MAXPTS,<max_points>
[,SWEEPS,<max_sweeps>]

* 4–channel oscilloscopes only

EXAMPLE (GPIB)

The following command defines Trace A to compute the summed average of Channel 1 using 5000 points over 200 sweeps:

```
CMD$="TA:DEF EQN,'AVGS(C1)',
MAXPTS, 5000,SWEEPS,200"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

FUNCTION_RESET, INR?

FUNCTION

DEFINE, DEF

Command/Query

Oscilloscopes fitted with the WP01 Option

DESCRIPTION

An oscilloscope fitted with the Waveform Processing option (WP01) accepts additional forms of the DEFINE command:

Processing Notation	
ABS	Absolute Value
AVGC	Continuous Average
AVGS	Summed Average
DERI	Derivative
ERES	Enhanced Resolution Filter
EXP	Exponential (power of e)
EXP10	Exponential (power of 10)
EXTR	Extrema
FLOOR	Floor (Extrema only)
INTG	Integral
LN	Logarithm base e
LOG10	Logarithm base 10
RESC	Rescale
ROOF	Roof (Extrema only)
SINX	Sin(X)/X interpolator
SQR	Square
SQRT	Square Root
ZOOMONLY	No Math
+	Identity or Add
–	Negation or Subtract
*	Multiply
/	Ratio
1/	Reciprocal

Key words	
BITS	Resolution enhancement bits (Enhanced Resolution only)
MAXPTS	Maximum number of points
SWEEPS	Maximum number of sweeps (Average and Extrema only)
WEIGHT	Weight (Continuous Average only)

COMMAND SYNTAX

<function>:DEFine EQN,'<equation>', MAXPTS,<max_points>, SWEEPS,<max_sweeps>,WEIGHT,<weight>, BITS,<bits>

<function> := {TA, TB, TC, TD}

<equation> := AVGS(<source>)	Summed Average
<equation> := AVGC(<source>)	Continuous Average
<equation> := <source>	Identity
<equation> := +<source>	Identity
<equation> := –<source>	Negation
<equation> := 1/<source>	Reciprocal
<equation> := <source> + <source>	Addition
<equation> := <source> – <source>	Subtraction
<equation> := <source> * <source>	Multiplication
<equation> := <source> / <source>	Ratio
<equation> := EXTR(<source>)	Extrema (R+F)
<equation> := FLOOR(EXTR(<source>))	Floor
<equation> := ROOF(EXTR(<source>))	Roof
<equation> := SINX(<source>)	Sin(X)/X interpolator
<equation> := SQR(<source>)	Square
<equation> := SQRT(<source>)	Square Root
<equation> := LN(<source>)	Logarithm base e
<equation> := LOG10(<source>)	Logarithm base 10
<equation> := EXP(<source>)	Power of e
<equation> := EXP10(<source>)	Power of 10
<equation> := INTG(<source> [{+,–}<addend>])	Integral
<equation> := DERI(<source>)	Derivative
<equation> := ABS(<source>)	Absolute Value

<equation> := ERES(<source>) Enhanced Resolution

<equation> := RESC([{+,–}] Rescale
[<multiplier>*]<source>
[{+,–}<addend>])

<addend> := 0.0 to 10^{15}

<multiplier>:=0.0 to 10^{15}

<source> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}

	L models	M models	Others
<max_points> :=	50000	50000	10000

<max_sweeps> := 1 to 1000000

<weight> := {1, 3, 7, 15, 31, 63, 127, 255, 511,1023}

<bits> := {0.5, 1.0, 1.5, 2.0, 2.5, 3.0}

Note: Space (blank) characters inside equations are optional.

QUERY SYNTAX

<function>:DEFINE?

Response format

<function>:DEFine EQN,'<equation>'[,MAXPTS,<max_points>] [,SWEEPS,<max_sweeps>][,WEIGHT,<weight>][,BITS,<bits>]

EXAMPLE (GPIB):

The following command defines Trace A to compute the product of Channel 1 and Channel 2, using a maximum of 10000 input points:

```
CMD$="TA:DEF EQN,'C1*C2', MAXPTS,10000"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

FUNCTION_RESET, INR?

* 4–channel oscilloscopes only

FUNCTION

DEFINE, DEF

Command/Query

Oscilloscopes fitted with the WP02 Option

DESCRIPTION

An oscilloscope fitted with the FFT option (WP02) accepts additional forms of the DEFINE command.

Notation	
AVGP	Power Average
BLHA	Blackman–Harris window
FLTP	Flat Top window
FFT	Fast Fourier Transform (complex result)
HAMM	Hamming window
HANN	von Hann window
IMAG	Imaginary part of complex result
MAG	Magnitude of complex result
PHASE	Phase angle (degrees) of complex result
PS	Power Spectrum
PSD	Power Density
REAL	Real part of complex result
RECT	Rectangular window
RESC	Rescale
WINDOW	FFT window function

COMMAND SYNTAX (FFT)

<function>:DEFine EQN,'<equation>', MAXPTS,<max_points>, WINDOW,<window_type>

<function> := {TA, TB, TC, TD}
<equation> := FFT(<source>)
<equation> := REAL(FFT(<source>))
<equation> := IMAG(FFT(<source>))
<equation> := MAG(FFT(<source>))
<equation> := PHASE(FFT(<source>))
<equation> := PS(FFT(<source>))
<equation> := PSD(FFT(<source>))
<source> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}
<window_type> := {RECT, HANN, HAMM, FLTP, BLHA}

	L models	M models	Others
<max_points> :=	50000	50000	10000

Note: The source waveform must be a time domain signal, single segment.

QUERY SYNTAX

<function>:DEFine?

Response Format

<function>:DEFine EQN,'<equation>',MAXPTS,<max_points>, WINDOW,<window_type>

EXAMPLE (GPIB)

The following command defines Trace A to compute the Power Spectrum of the FFT of Channel 1. A maximum of 1000 points will be used for the input. The window function is Rectangular.

```
CMD$="TA:DEF EQN,'PS(FFT(C1))',
MAXPTS,1000,WINDOW,RECT"
CALL IBWRT(SCOPE%,CMD$)
```

COMMAND SYNTAX

(FFT Power Average)

<function>:DEFine EQN,'<equation>', SWEEPS,<max_sweeps>
<equation> := MAG(AVGP(<source>))
<equation> := PS(AVGP(<source>))
<equation> := PSD(AVGP(<source>))
<source> := {TA, TB, TC, TD}
<max_sweeps> := 1 to 50000

Note: The source waveform must be another function defined as a Fourier transform.

(Rescale)

<equation> := RESC([{+,–}] Rescale
[<multiplier>*]<source>
[{+,–}<addend>])
<addend> := 0.0 to 10^{15}
<multiplier>:=0.0 to 10^{15}
<source> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}

QUERY SYNTAX

<function>:DEFine?

Response Format

<function>:DEFine EQN,'<equation>',SWEEPS,<max_sweeps>

EXAMPLE (GPIB)

The following command defines Trace B to compute the Power Spectrum of the Power Average of the FFT being computed by the Trace A, over a maximum of 244 sweeps.

```
CMD$="TB:DEF EQN,'PS(AVGP(TA))',SWEEPS,244"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

FUNCTION_RESET, INR?

* 4–channel oscilloscopes only

MISCELLANEOUS

DELETE_FILE, DELF

Command

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The DELETE_FILE command deletes files from mass storage.

COMMAND SYNTAX

DELete_file DISK,<dev>,FILE, '<filename>'

<dev> := {CARD, FLPY}

<filename> := an alphanumeric string of up to 8 characters, followed by a dot and an extension of up to 3 characters.

EXAMPLE (GPIB)

The following command deletes a front–panel setup from the memory card:

```
CMD$="DELF DISK,CARD,FILE,'P001.PNL';
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DIRECTORY_LIST, FORMAT_CARD, FORMAT_FLOPPY

MISCELLANEOUS

DIRECTORY_LIST?, DIR?

Query

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The DIRECTORY_LIST? query produces a directory listing of a memory card or a floppy disk. The response consists of a double–quoted string containing a DOS–like listing of the directory. If no card or floppy is present, or if it is not formatted, the string will be empty.

QUERY SYNTAX

DIRectory_list? DISK,<dev>

<dev> := {CARD, FLPY}

Response format

DIRectory_list? DISK,<dev> "<directory>"

<directory> := a variable length string detailing the file content of the memory card or the floppy disk.

EXAMPLE (GPIB)

The following code asks for a listing of the directory of the memory card:

```
CMD$="DIR? DISK,CARD": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD (SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
DIR "
Memory Card of 15–JAN–1992 12:10:40
                SC1  001 20361 19–JAN–1992  16:33:06
                SC2  001 20361 1 9–JAN–1992 16:34:32
2 File(s) 84992 bytes free
"
```

DISPLAY

DISPLAY, DISP

Command/Query

DESCRIPTION

The DISPLAY command controls the display screen of the oscilloscope. When the user is remotely controlling the oscilloscope and does not need to use the display, it may be useful to switch off the display via the DISPLAY OFF command. This improves instrument response time since the waveform graphic generation procedure is suppressed.

The response to the DISPLAY? query indicates the display state of the oscilloscope.

Note: When the display has been set to OFF, the real–time clock and the message field are updated. However, the waveforms and associated texts remain unchanged.

COMMAND SYNTAX

DISPlay <state>

<state> := {ON, OFF}

QUERY SYNTAX

DISPlay?

Response Format

DISPlay <state>

EXAMPLE (GPIB)

The following instruction turns off the display generation:

CMD$="DISP OFF": CALL IBWRT(SCOPE%,CMD$)

DISPLAY

DUAL_ZOOM, DZOM

Command/Query

DESCRIPTION

By setting DUAL_ZOOM ON, the horizontal magnification and positioning controls apply to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The DUAL_ZOOM? query indicates whether multiple zoom is enabled or not.

Note: This command has the same effect as MULTI_ZOOM.

COMMAND SYNTAX

Dual_ZOoM <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

Dual_ZOoM?

Response format

Dual_ZOoM <mode>

EXAMPLE (GPIB)

The following example turns dual zoom on:

CMD$="DZOM ON": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

HOR_MAGNIFY, HOR_POSITION, MULTI_ZOOM

STATUS

*ESE

Command/Query

DESCRIPTION

The *ESE command sets the standard Event Status Enable register (ESE). This command allows one or more events in the ESR register to be reflected in the ESB summary message bit (bit 5) of the STB register. For an overview of the ESB defined events refer to the ESR table on page 5–47.

The *ESE? query reads the contents of the ESE register.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

*ESE <value>

<value> := 0 to 255

QUERY SYNTAX

*ESE?

Response format

*ESE <value>

EXAMPLE (GPIB)

The following command allows the ESB bit to be set if a user request (URQ bit 6, i.e. decimal 64) and/or a device dependent error (DDE bit 3, i.e. decimal 8) occurs. Summing these values yields the ESE register mask 64+8=72.

```
CMD$="*ESE 72": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*ESR?

STATUS

***ESR?**
Query

DESCRIPTION

The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7. Refer to the table on page 5–47 for an overview of the ESR register structure.

QUERY SYNTAX

*ESR?

Response format

*ESR <value>

<value> := 0 to 255

EXAMPLE (GPIB)

The following instruction reads and clears the contents of the ESR register:

```
CMD$="*ESR?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:
*ESR 0

RELATED COMMANDS

ALL_STATUS?, *CLS, *ESE

Bit	Bit Value	Bit Name		Description	Note
15..8			0	Reserved by IEEE 488.2	
7	128	PON	1	a Power off–to–ON transition has occurred	(1)
6	64	URQ	1	a User ReQuest has been issued	(2)
5	32	CME	1	a CoMmand parser Error has been found	(3)
4	16	EXE	1	an Execution Error has been detected	(4)
3	8	DDE	1	a Device Specific Error has occurred	(5)
2	4	QYE	1	a QueRy Error has occurred	(6)
1	2	RQC	0	The instrument never requests bus control	(7)
0	1	OPC	0	The OPeration Complete bit is not used	(8)

Standard Event Status Register (ESR)

Notes:

(1) The Power On (PON) bit is always turned on (1) when the unit is powered up.

(2) The User Request (URQ) bit is set true (1) when a soft key is pressed. An associated register URR identifies which key was selected. For further details refer to the URR? query.

(3) The CoMmand parser Error bit (CME) is set true (1) whenever a command syntax error is detected. The CME bit has an associated CoMmand parser Register (CMR) which specifies the error code. Refer to the query CMR? for further details.

(4) The EXecution Error bit (EXE) is set true (1) when a command cannot be executed due to some device condition (e.g. oscilloscope in local state) or a semantic error.The EXE bit has an associated Execution Error Register (EXR) which specifies the error code. Refer to query EXR? for further details.

(5) The Device specific Error (DDE) is set true (1) whenever a hardware failure has occurred at power–up or execution time such as a channel overload condition, a trigger or a time–base circuit defect. The origin of the failure may be localized via the DDR? or the self test *TST? query.

(6) The Query Error bit (QYE) is set true (1) whenever (a) an attempt is being made to read data from the Output Queue when no output is either present or pending, (b) data in the Output Queue has been lost, (c) both output and input buffers are full (deadlock state), (d) an attempt is made by the controller to read before having sent an <END>, (e) a command is received before the response to the previous query was read (output buffer flushed).

(7) The ReQuest Control bit (RQC) is always false (0) since the oscilloscope has no GPIB controlling capability.

(8) The OPeration Complete bit (OPC) is set true (1) whenever *OPC has been received since commands and queries are strictly executed in sequential order. The oscilloscope starts processing a command only once the previous command has been entirely executed.

STATUS

EXR?
Query

DESCRIPTION

The EXR? query reads and clears the contents of the EXecution error Register (EXR). The EXR register specifies the type of the last error detected during execution. Refer to the table on page 5–49 for further details.

QUERY SYNTAX

EXR?

Response format

EXR <value>

<value> := 21 to 64

EXAMPLE (GPIB)

The following instruction reads the contents of the EXR register:

```
CMD$="*EXR?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message (if no fault)
EXR 0

RELATED COMMANDS

ALL_STATUS?, *CLS

Value	Description
21	Permission error. The command cannot be executed in local mode.
22	Environment error. The instrument is not configured to correctly process a command. For instance, the oscilloscope cannot be set to RIS at a slow time base.
23	Option error. The command applies to an option which has not been installed.
24	Unresolved parsing error.
25	Parameter error. Too many parameters specified.
26	Non–implemented command.
30	Hex data error. A non–hexadecimal character has been detected in a hex data block.
31	Waveform error. The amount of data received does not correspond to descriptor indicators.
32	Waveform descriptor error. An invalid waveform descriptor has been detected.
33	Waveform text error. A corrupted waveform user text has been detected.
34	Waveform time error. Invalid RIS or TRIG time data has been detected.
35	Waveform data error. Invalid waveform data have been detected.
36	Panel setup error. An invalid panel setup data block has been detected.
50	No mass storage present when user attempted to access it. @
51	Mass storage not formatted when user attempted to access it. @
53	Mass storage was write protected when user attempted to create a file, to delete a file, or to format the device. @
54	Bad mass storage detected during formatting. @
55	Mass storage root directory full. LECROY_1.DIR subdirectory cannot be created. @
56	Mass storage full when user attempted to write to it. @
57	Mass storage file sequence numbers exhausted (999 reached). @
58	Mass storage file not found. @
59	Attempt to retrieve a file from mass storage that is neither a waveform nor a panel. @
60	Mass storage file is write protected (DOS Read/Only attribute). @
61	Mass storage filename not DOS compatible, or waveform filename begins with an "A". @
62	Cannot write on mass storage because filename already exists. @

@ only oscilloscopes fitted with the memory card (MC01) or floppy disk (FD01) options

Execution Error Status Register Structure (EXR)

MISCELLANEOUS

FORMAT_CARD, FCRD

Command/Query

Oscilloscopes fitted with the MC01 option

DESCRIPTION

The FORMAT_CARD command formats the memory card according to the PCMIA/JEIDA standard with a DOS partition.

The FORMAT_CARD? query returns the status of the card.

COMMAND SYNTAX

Format_CaRD

QUERY SYNTAX

Format_CaRD?

Response format

Format_CaRD <card_status>[,<read/write>,<free_space>,
<card_size>,<battery_status>]

<card_status> := {NONE BAD, BLANK, DIR_MISSING, OK}
<read/write> := {WP, RW}
<free_space> := a decimal number giving the number of bytes still available on the card.
<card_size> := a decimal number giving the total number of bytes on the card.
<battery_status> := {BAT_OK, BAT_LOW, BAT_BAD}

Notation	
BAD	Bad card after formatting
BAT_BAD	Bad battery or no battery
BAT_LOW	The battery should be replaced
BAT_OK	The battery is in order
BLANK	Empty card
DIR_MISSING	No "LECROY 1_DIR" subdirectory present. It will be automatically created with the next "store" command
NONE	No card
OK	The card is correctly formatted
RW	Read/Write authorized
WP	Write protected

EXAMPLE (GPIB)

The following code will first format a memory card and then verify its status.

```
CMD$="FCRD": CALL IBWRT(SCOPE%,CMD$)
CMD$="FCRD?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message

```
FCRD OK,RW,130048,131072,BAT_OK
```

RELATED COMMANDS

DIRECTORY_LIST

MISCELLANEOUS

FORMAT_FLOPPY, FFLP

Command/Query

Oscilloscopes fitted with the FD01 option

DESCRIPTION

The FORMAT_FLOPPY command formats a floppy disk in the Double Density or High Density format.

The FORMAT_FLOPPY? query returns the status of the floppy disk.

COMMAND SYNTAX

Format_FLoPpy [{DD,HD}]

QUERY SYNTAX

Format_FLoPpy?

Response format

Format_FLoPpy <floppy_status>[,<read/write>,<free_space>, <floppy_size>]

<floppy_status> := {NONE, BAD, BLANK, DIR_MISSING, OK}
<read/write> := {WP, RW}
<free_space> := a decimal number giving the number of bytes still available on the floppy.
<floppy_size> := a decimal number giving the total number of bytes on the floppy.

Notation	
BAD	Bad floppy after formatting
BLANK	Empty floppy
DD	Double Density 720 kB formatted
DIR_MISSING	No "LECROY 1_DIR" subdirectory present. It will be automatically created with the next "store" command
HD	High Density 1.44 MB formatted
NONE	No floppy
OK	The floppy is correctly formatted
RW	Read/Write authorized
WP	Write protected

EXAMPLE (GPIB)

The following code will first format a floppy disk in the double density (720 kB) format and then verify its status:

```
CMD$="FFLP DD": CALL IBWRT(SCOPE%,CMD$)
CMD$="FFLP?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
FFLP OK,RW,728064,737280
```

RELATED COMMANDS

DIRECTORY_LIST

FUNCTION

FUNCTION_RESET, FRST

Command

DESCRIPTION

The FUNCTION_RESET command resets a waveform processing function. The number of sweeps will be reset to zero and the process restarted.

COMMAND SYNTAX

<function>:Function_ReSeT

EXAMPLE (GPIB)

<function> := {TA, TB, TC, TD}

Assuming that Trace A (TA) has been defined as the summed average of Channel 1, the following example will restart the averaging process:

```
CMD$="TA:FRST": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, INR?

DISPLAY

GRID

Command/Query

DESCRIPTION

The GRID command specifies whether the grid should be displayed in single, dual or quad mode.

The GRID? query returns the grid mode currently in use.

COMMAND SYNTAX

GRID <grid>

<grid> := {SINGLE, DUAL, QUAD*}

QUERY SYNTAX

GRID?

Response Format

GRID <grid>

EXAMPLE (GPIB)

The following command sets the screen display to dual grid mode:

CMD$="GRID DUAL": CALL IBWRT(SCOPE%,CMD$)

* 4–channel oscilloscopes only

HARD COPY

HARDCOPY_SETUP, HCSU

Command/Query

DESCRIPTION

The HARDCOPY_SETUP command configures the instrument's hard copy driver. The command enables the user to specify the device type and transmission mode of the hard–copy unit connected to the oscilloscope.

The command allows one or more individual settings to be changed by specifying the appropriate keyword(s) together with the new value(s).

Notation			
DEV	device	HPDJ	HP DeskJet
PENS	plot pens	HPPJ	HP PaintJet
PFEED	page feed	HPLJ	HP LaserJet
PORT	transmission mode	HPTJ	HP ThinkJet
PSIZE	paper size	HP7470A	HP7470 plotter
TIFF	TIFF format	HP7550A	HP7550 plotter
TIFFCOMP	Compressed TIFF	EPSON	Epson FX80

COMMAND SYNTAX

HardCopy_SetUp DEV,<device>, PORT,<port>, PFEED,<page_feed>, PENS,<plot_pens>, PSIZE<paper_size>

<device> := {HPDJ, HPPJ, HPTJ, HPLJ, HP7470A, HP7550A, EPSON, TIFF, TIFFCOMP}

<port> := {GPIB, RS, CENT, FLPY, CARD, PRT}

<page_feed> := {ON, OFF}

<plot_pens> := 1 to 8

<paper_size> := {A5, A4}

Note: Parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be given in any order and may be restricted to those variables to be changed.

QUERY SYNTAX

HardCopy_SetUp?

Response format

HardCopy_SetUp DEV,<device>, PORT,<port>, PFEED,<page_feed>, PENS,<plot_pens>, PSIZE,<paper_size>

EXAMPLE (GPIB)

This example selects an EPSON printer to be connected via the RS232 port:

```
CMD$="HCSU PORT,RS,DEV,EPSON"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HARDCOPY_TRANSMIT, SCREEN_DUMP

HARD COPY

HARDCOPY_TRANSMIT, HCTR

Command

DESCRIPTION

The HARDCOPY_TRANSMIT command sends a string of ASCII characters without modification to the hard–copy unit. This allows the user to control the hard–copy unit by sending device specific control character sequences. It also allows the user to place additional text on a screen dump for documentation purposes. This command accepts the escape sequence "\ddd" like those described under the command COMM_RS232 (see page 5–23). Before sending the string to the hard–copy unit the escape sequence is converted to the ASCII character code.

COMMAND SYNTAX

HardCopy_TRansmit '<string>'
<string> := Any sequence of ASCII or escaped characters.

EXAMPLE (GPIB)

The following code sends documentation data to a printer:

```
CMD$="HCTR 'Data from Oct.15\r\n'"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HARDCOPY_SETUP, SCREEN_DUMP

DISPLAY

HOR_MAGNIFY, HMAG

Command/Query

DESCRIPTION

The HOR_MAGNIFY command horizontally expands the selected expansion trace by a specified factor. Magnification factors which are not within the range of permissible values will be rounded to the closest legal value.

If multiple zoom is enabled, the magnification factor for all expansion traces is set to the specified factor. If the specified factor is too large for any of the expanded traces (depending on their current source), it is reduced to an acceptable value and only then applied to the traces.

The VAB bit (bit 2) in the STB register (see table on page 5–107) is set if a factor outside the legal range is specified.

The HOR_MAGNIFY query returns the current magnification factor for the specified expansion function.

COMMAND SYNTAX

<exp_trace>:Hor_MAGnify <factor>

<exp_trace> := {TA, TB, TC, TD}
<factor> := 1 up to 20000

<factor> :=	L models	M models	Others
	20000	1000	200

QUERY SYNTAX

<exp_source>:Hor_MAGnify?

Response Format

<exp_source>:Hor_MAGnify <factor>

EXAMPLE (GPIB)

The following example horizontally magnifies Trace A (TA) by a factor of 5:

```
CMD$="TA:HMAG 5": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DUAL_ZOOM, MULTI_ZOOM

DISPLAY

HOR_POSITION, HPOS

Command/Query

DESCRIPTION

The HOR_POSITION command horizontally positions the geometric center of the intensified zone on the source trace. Allowed positions range from division 0 through 10. If the source trace was acquired in sequence mode, horizontal shifting will only apply to a single segment at a time.

If the multiple zoom is enabled, the difference between the specified and the current horizontal position of the specified trace is applied to all expanded traces. If this would cause the horizontal position of any expanded trace to go outside the left or right screen boundaries, the difference of positions is adapted and then applied to the traces.

If the sources of expanded traces are sequence waveforms, and the multiple zoom is enabled, the difference between the specified and the current segment of the specified trace is applied to all expanded traces. If this would cause the segment of any expanded trace to go outside of the range of the number of segments of sources, the difference is adapted and then applied to the traces.

The VAB bit (bit 2) in the STB register (see table on page 5–107) is set if a value outside the legal range is specified.

The HOR_POSITION query returns the position of the geometric center of the intensified zone on the source trace.

Note: Segment number 0 has the special meaning "Show All Segments Unexpanded".

COMMAND SYNTAX

<exp_trace>:Hor_POSition <hor_position>,

<exp_trace> := {TA, TB, TC, TD}
<hor_position> := 0 to 10 DIV

 :=

L models	M models	Others
2000	200	50

Note 1: The suffix DIV is optional.

Note 2: The segment number is only relevant for waveforms acquired in sequence mode; it is ignored in single waveform acquisitions. When the segment number is set to 0, all segments will be shown.

QUERY SYNTAX

<exp_trace>:Hor_POSition?

Response Format

<exp_trace>:Hor_POSition <hor_position>,[]

Note: The segment number is only given for sequence waveforms.

EXAMPLE (GPIB)

The following example positions the center of the intensified zone on the trace currently viewed by Trace A (TA) at division 3:

CMD$="TA:HPOS 3": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

DUAL_ZOOM, MULTI_ZOOM

MISCELLANEOUS

*IDN?

Query

DESCRIPTION

The *IDN? query is used for identification purposes. The response consists of four different fields providing information on the manufacturer, the scope model, the serial number and the firmware revision level.

QUERY SYNTAX

*IDN?

Response format

*IDN LECROY,<model>,<serial_number>,<firmware_level>

<model> := 5–character model identifier

<serial_number> := an 9–digit decimal code (93XXXXXXX)

<firmware_level> := 2 digits giving the release level followed by a period and a 1–digit update level (xx.y)

EXAMPLE (GPIB)

This example issues an identification request to the scope:

```
CMD$="*IDN?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
*IDN  LECROY,9314L_,931401000,6
```

STATUS

INE

Command/Query

DESCRIPTION

The INE command sets the Internal state change Enable register (INE). This command allows one or more events in the INR register to be reflected in the INB summary message bit (bit 0) of the STB register. For an overview of the INR defined events, refer to the table on page 5–64.

The INE? query reads the contents of the INE register.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

INE <value>

<value> := 0 to 65 535

QUERY SYNTAX

INE?

Response format

INE <value>

EXAMPLE (GPIB)

The following command allows the INB bit to be set whenever a screen dump has finished (bit 1, i.e. decimal 2) and/or a waveform has been acquired (bit 0, i.e. decimal 1). Summing these two values yields the INE mask 2+1=3.

```
CMD$="INE 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INR?

STATUS

INR?

Query

DESCRIPTION

The INR? query reads and clears the contents of the INternal state change Register (INR). The INR register (see table below) keeps track of the completion of various internal operations and state transitions.

QUERY SYNTAX

INR?

Response format

INR <state>

<state> := 0 to 65535

EXAMPLE (GPIB)

The following instruction reads the contents of the INR register:

CMD$="INR?": CALL IBWRT(SCOPE%,CMD$)

Response message:

INR 1026

i.e. waveform processing in Function E and a screen dump have both terminated.

RELATED COMMANDS

ALL_STATUS?, *CLS, INE

Bit	Bit Value		Description
15...13		0	Reserved for future use
12	4096	1	Pass/Fail test detected desired outcome
11	2048	1	Waveform processing has terminated in Trace D
10	1024	1	Waveform processing has terminated in Trace C
9	512	1	Waveform processing has terminated in Trace B
8	256	1	Waveform processing has terminated in Trace A
7	128	1	A memory card or floppy exchange has been detected
6	64	1	Memory card or floppy has become full in "Autostore Fill" mode
5	32	0	Reserved
4	16	1	A segment of a sequence waveform has been acquired
3	8	1	A time–out has occurred in a data block transfer
2	4	1	A return to the local state is detected
1	2	1	A screen dump has terminated
0	1	1	A new signal has been acquired

Internal State Register Structure (INR)

WAVEFORM TRANSFER

INSPECT?, INSP?

Query

DESCRIPTION

The INSPECT? query allows the user to read parts of an acquired waveform in intelligible form. The command is based on the explanation of the format of a waveform given by the template (use the query TEMPLATE? to obtain an up–to–date copy). Each logical block of a waveform may be inspected by giving its name (e.g. TRIGTIME as mentioned in the template) enclosed in quotes as the first (string) parameter.

The special logical block named WAVEDESC may also be inspected in more detail. By giving the name of a variable in the block WAVEDESC enclosed in quotes as the first (string) parameter, it is possible to inspect only the actual value of that variable.

Notation	
BYTE:	raw data as integers (truncated to 8 m.s.b.*)
FLOAT:	normalized data (gain, offset applied) as floating point numbers (gives measured values in volts or appropriate units)
WORD:	raw data as integers (truncated to 16 m.s.b.*)

* *most significant bits*

QUERY SYNTAX

<trace>:INSPect? '<string>'[,<data_type>]

<trace> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}
<string> := a valid name of a logical block or a valid name of a variable contained in block WAVEDESC (see the command TEMPLATE and Section 6).

<data_type> := {BYTE, WORD, FLOAT}

Note: The optional parameter <data_type> applies only for inspecting the data arrays. It selects the representation of the data. The default <data type> is FLOAT.

Response format

<trace>:INSPect "<string>"

<string> := a string giving name(s) and value(s) of a logical block or a variable.

* 4–channel oscilloscopes only

EXAMPLES (GPIB)

1) The following command reads the value of the time base at which the last waveform in Channel 1 was acquired:

```
CMD$:="C1:INSP? 'TIMEBASE'"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:
C1:INSP "TIMEBASE: 500 US/DIV"

2) The following command reads the entire contents of the waveform descriptor block:

```
CMD$ = "C1:INSP? 'WAVEDESC'"
```

RELATED COMMANDS

TEMPLATE, WAVEFORM_SETUP

DISPLAY

INTENSITY, INTS

Command/Query

DESCRIPTION

The INTENSITY command sets the intensity level of the grid or the trace/text.

The intensity level is expressed as a percentage (PCT). A level of 100 PCT corresponds to the maximum intensity while a level of 0 PCT sets the intensity to its minimum value.

The response to the INTENSITY? query indicates the grid and trace intensity levels.

COMMAND SYNTAX

INTenSity,GRID,<value>,TRACE,<value>

<value> := 0 to 100 PCT

Note 1: Parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be given in any order and may be restricted to those variables to be changed.

Note 2: The suffix PCT is optional.

QUERY SYNTAX

INTenSity?

Response Format

INTenSity,TRACE,<value>,GRID,<value>

EXAMPLE (GPIB)

The following instruction enables remote control of the intensity and changes the grid intensity level to 75%:

```
CMD$="INTS,GRID,75"
CALL IBWRT(SCOPE%,CMD$)
```

ACQUISITION

INTERLEAVED, ILVD

Command/Query

DESCRIPTION

The INTERLEAVED command enables or disables random interleaved sampling (RIS) for timebase settings where both single shot and RIS mode are available.

Memory Size	50K or 1M	10K
Selectable RIS/Single Shot	50 ns/div to 20 µs/div	50 ns/div to 5 µs/div

RIS is not available for sequence mode acquisitions.

The response to the INTERLEAVED? query indicates whether the oscilloscope is in RIS mode.

COMMAND SYNTAX

InterLeaVeD <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

InterLeaVeD?

Response Format

InterLeaVeD <mode>

EXAMPLE

The following command instructs the oscilloscope to use RIS mode:

```
CMD$ = "ILVD ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TIME_DIV, TRIG_MODE, MEMORY_SIZE

STATUS

***IST?**
Query

DESCRIPTION

The *IST? (Individual STatus) query reads the current state of the IEEE 488.1 defined "ist" local message. The "ist" individual status message is the status bit sent during a parallel poll operation.

QUERY SYNTAX

*IST?

Response format

*IST <value>

<value> := 0 or 1

EXAMPLE (GPIB)

The following command reads the contents of the IST bit:

```
CMD$ = "*IST?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message
*IST 0

RELATED COMMANDS

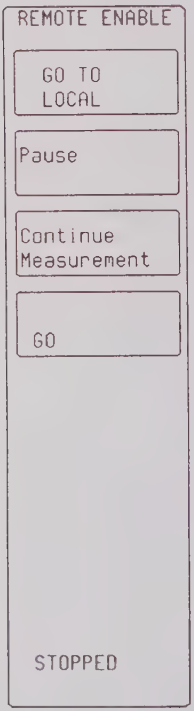
*PRE

DISPLAY

KEY

Command

DESCRIPTION



The KEY command is used to display a string in the menu field next to buttons 2 – 6. The string may consist of up to 2 lines of 13 characters (see menu).

Text assigned to the menu buttons will disappear on the next transition to local but reappear when the instrument is switched back into the remote state. Text is cleared at power up, when the instrument is reset, or if an empty string is assigned to a location (e.g. KEY 2,' ').

Pressing any one of the menu buttons while in remote mode causes the User Request status Register (URR) and the URQ bit of the Event Status Register to be set. This can generate an SRQ provided that the service request mechanism has been enabled.

Note: This command can be executed in both local and remote modes (although the text will only appear when in remote mode).

COMMAND SYNTAX

KEY <button>,'<string>','<string>'

<button> := 2 to 6
<string> := up to two 13–character strings (any ASCII code)

EXAMPLE (GPIB)

The example menu shown in the figure on page 5–70 was created by issuing the following series of KEY commands:

```
KEY 2, 'Pause';
KEY 3, 'Continue','Measurement';
KEY 4, ' ', ' GO'
```

RELATED COMMANDS

URR?

ACQUISITION

MEMORY_SIZE, MSIZ

Command (M and L–models only)/Query

DESCRIPTION

The MEMORY_SIZE command is accepted only by L–models. It allows selection of the maximum memory length used for acquisition: 10000, 50000, or 1000000 data points. Reducing the number of data points results in faster throughput.

The MEMORY_SIZE? query returns the current maximum memory length used to capture waveforms.

COMMAND SYNTAX

Memory_SIZe <size>

<size> := {10K, 50K, 1M}

QUERY SYNTAX

Memory_SIZe?

Response Format

Memory_SIZe <size>

EXAMPLE

The following command will set the oscilloscope to acquire at most 10000 data samples per single shot or RIS acquisition:

```
CMD$="MSIZ 10K" :
CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

MESSAGE, MSG

Command/Query

DESCRIPTION

The MESSAGE command displays a string of characters in the Message Field above the grid. The string may be up to 48 characters in length. The string is displayed as long as the instrument is in remote mode and no internal status message is generated. Turning the oscilloscope back to local mode deletes the message. After the next transition from local to remote the message will be redisplayed. The message is cleared at power up, when the instrument is reset, or if an empty string is sent (MSG " ").

The MESSAGE? query allows the user to read the last message which was sent.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

MeSsaGe '<string>'

<string> := a string of max. 48 characters

QUERY SYNTAX

MeSsaGe?

Response Format

MeSsaGe "<string>"

EXAMPLE (GPIB)

The following code causes the message "*Connect Probe 1*" to appear in the message field:

```
CMD$="MSG "*Connect Probe 1*""
CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

MULTI_ZOOM, MZOM

Command/Query

DESCRIPTION

By setting MULTI_ZOOM ON, the horizontal magnification and positioning controls apply to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The MULTI_ZOOM? query indicates whether multiple zoom is enabled or not.

Note: This command has the same effect as DUAL_ZOOM.

COMMAND SYNTAX

Multi_ZOoM <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

Multi_ZOoM?

Response format

Multi_ZOoM <mode>

EXAMPLE (GPIB)

The following example turns the multiple zoom on:

CMD$="MZOM ON": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

HOR_MAGNIFY, HOR_POSITION, DUAL_ZOOM

ACQUISITION

OFFSET, OFST

Command/Query

DESCRIPTION

The OFFSET command allows the vertical offset of the specified input channel to be adjusted.

The maximum ranges depend on the fixed sensitivity setting as follows:

2–10 mV/div	± 120 mV offset
20–200 mV/div	± 1.2 V offset
0.5 – 5 V/div	± 24 V offset

If an out–of–range value is entered, the oscilloscope is set to the closest possible value and the VAB bit (bit 2) in the STB register is set.

Note: The probe attenuation factor is not taken into account for adjusting the offset.

The OFFSET? query returns the DC offset value of the specified channel.

COMMAND SYNTAX

<channel>:OFfSeT <offset>

<channel> := {C1, C2, C3*, C4*}
<offset> := – 24V to 24V (maximum range)

Note: The suffix V is optional.

QUERY SYNTAX

<channel>:OFfSeT?

Response format

<channel>:OFfSeT <offset>

EXAMPLE (GPIB)

The following command sets the offset of Channel 2 to –3 V:

```
CMD$="C2:OFST –3V": CALL IBWRT(SCOPE%,CMD$)
```

* 4–channel oscilloscopes only

STATUS

***OPC**

Command/Query

DESCRIPTION

The *OPC (OPeration Complete) command sets the OPC bit (bit 0) in the standard Event Status Register (ESR) to true. This command has no other effect on the operation of the oscilloscope as the instrument starts parsing a command or query only after it has completely processed the previous command or query.

The *OPC? query always responds with the ASCII character "1" as the oscilloscope responds to the query only once the previous command has been entirely executed.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

*OPC

QUERY SYNTAX

*OPC?

Response format

*OPC 1

RELATED COMMANDS

*WAI

MISCELLANEOUS

*OPT?

Query

DESCRIPTION

The *OPT? query identifies oscilloscope options, i.e. additional firmware or hardware options. The response consists of a series of response fields listing all the installed options.

QUERY SYNTAX

*OPT?

Response format

*OPT <option_1>,<option_2>,..,<option_N>

<option_i> := character data

Note: If no option is present, the character 0 will be returned.

EXAMPLE (GPIB)

This example queries the installed options:

CMD$="*OPT?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$

Response message:

*OPT 0

If the waveform processing options WP01 and WP02 are installed, the response message is:

*OPT WP01,WP02

SAVE/RECALL SETUP

PANEL_SETUP, PNSU

Command/Query

DESCRIPTION

The PANEL_SETUP command complements the *SAV/*RST commands. The PANEL_SETUP command allows panel setups to be archived in encoded form on external storage media.

Only setup data read by the PNSU? query may be recalled into the oscilloscope. A panel setup error (see table on page 5–49) will be generated if the setup data block contains invalid data.

Note: The communication parameters (those modified by commands CFMT, CHDR, CHLP, CORD and WFSU) and the enable registers associated with the status reporting system (SRE, PRE, ESE, INE) are **not** *saved by this command.*

COMMAND SYNTAX

PaNel_SetUp <setup>

<setup> := A setup block previously read by PNSU?

QUERY SYNTAX

PaNel_SetUp?

Response syntax

PaNel_SetUp <setup>

EXAMPLE (GPIB)

1. The following instruction saves the instrument's current panel setup in the file PANEL.SET:

```
FILE$ = "PANEL.SET": CMD$ = "PNSU?"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRDF(SCOPE%,FILE$)
```

2. The following command recalls the front–panel setup stored previously in the file PANEL.SET into the oscilloscope:

```
CALL IBWRTF(SCOPE%,FILE$)
```

RELATED COMMANDS

*RCL, *SAV

MISCELLANEOUS

PARAMETER_CLR, PACL

Command

DESCRIPTION

The PARAMETER_CLR command clears all the current parameters from the 5–line list used in the Custom and Pass/Fail modes.

Note: This command has the same effect as the command PASS_FAIL_CONDITION, given without any arguments.

COMMAND SYNTAX

PArameter_CleaR

RELATED COMMANDS

PARAMETER_DELETE, PARAMETER_VALUE, PASS_FAIL_CONDITION

MISCELLANEOUS

PARAMETER_DELETE, PADL

Command

DESCRIPTION

The PARAMETER_DELETE command deletes a parameter at a specified line from the list of parameters used in the Custom and Pass/Fail modes.

Notation		
1	line 1	of Custom or Pass/Fail display
2	line 2	of Custom or Pass/Fail display
3	line 3	of Custom or Pass/Fail display
4	line 4	of Custom or Pass/Fail display
5	line 5	of Custom or Pass/Fail display

COMMAND SYNTAX

PArameter_DeLete <line>

<line> := {1, 2, 3, 4, 5}

Note: This command has the same effect as the command PASS_FAIL_CONDITION <line>, given without any further arguments.

EXAMPLE (GPIB)

The following command deletes the third test condition in the list:

```
CMD$="PADL 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PARAMETER_CLR, PARAMETER_VALUE, PASS_FAIL_CONDITION

MISCELLANEOUS

PARAMETER_STATISTICS?, PAST?

Query

DESCRIPTION

The PARAMETER_STATISTICS? query returns the current values of statistics for the specified pulse parameter mode and the result type, for all five lines of the pulse parameters display.

Notation	
AVG	average
CUST	custom parameters
HIGH	highest value
HPAR	horizontal standard parameters
LOW	lowest value
PARAM	parameter definition for each line
SIGMA	sigma (standard deviation)
SWEEPS	number of sweeps accumulated for each line
VPAR	vertical standard parameters

QUERY SYNTAX

PArameter_STatistics? <mode>, <result>

<mode> := {CUST, HPAR, VPAR}

<result> := {AVG. LOW, HIGH, SIGMA, SWEEPS, PARAM}

Note: If keyword PARAM is specified, the query returns the list of the five pairs <parameter_name>,<source>.

EXAMPLE (GPIB)

The following query reads the average values of the five standard vertical parameters:

```
CMD$="PAST VPAR, AVG": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$): PRINT RD%
```

Response format

PAST VPAR, AVG, 13V, 26V, 47V, 1V, 0V

RELATED COMMANDS

PARAMETER_VALUE?

CURSOR

PARAMETER_VALUE?, PAVA?

Query

DESCRIPTION

The PARAMETER_VALUE? query returns the current value(s) of the pulse waveform parameter(s) and mask tests for the specified trace. Traces do not need to be displayed or selected to obtain the values measured by the pulse parameters or mask tests.

Pulse parameters cannot be evaluated on waveforms composed of segments acquired in sequence mode, or on waveforms with more than 50000 points. However pulse parameters may be applied to individual segments if they are singled out using the expansion function. Similarly, mask tests cannot be evaluated for sequence waveforms.

Parameter Names

ALL	all parameters	FREQ	frequency	OVSP	positive overshoot
AMPL	amplitude	FRST	first point	PER	period
AREA	area	LAST	last point	PKPK	peak–to–peak
BASE	base	MAX	maximum	PNTS	points
CYCL	cycles	MEAN	mean	RISE	risetime
DLY	delay	MEDI	median value	RMS	root mean square
DUTY	duty cycle	MIN	minimum	SDEV	standard deviation
FALL	falltime	OVSN	negative overshoot	TOP	top
				WID	width

Parameter Computation States

AV	averaged over several (up to 100) periods
GT	greater than given value
IV	invalid value (insufficient data provided)
LT	less than given value
NP	no pulse waveform
OF	signal partially in overflow
OK	deemed to be determined without problem
OU	signal partially in overflow and underflow
PT	window has been period truncated
UF	signal partially in underflow

Mask Test Names

INSIDE	all points of waveform inside mask (TRUE = 1, FALSE = 0)	OUTSIDE	some points of waveform outside mask (TRUE = 1, FALSE = 0)

Old Mask Test Names

ALLI	all points of waveform inside mask (TRUE = 1, FALSE = 0)	ANYO	some points of waveform outside mask (TRUE = 1, FALSE = 0)

QUERY SYNTAX

<trace>:PArameter_VAlue? [<parameter>, ... <parameter>]

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}
<parameter> := {ALL, AMPL, AREA, BASE, CYCL, DLY, DUTY, FALL, FREQ, FRST, LAST, MAX, MEAN, MEDI, MIN, OVSN, OVSP, PER, PKPK, PNTS, RISE, RMS, SDEV, TOP, WID}

Alternative forms of query for mask tests:

<trace>:PArameter_VAlue? <old_mask_test>
<trace>:PArameter_VAlue? <mask_test>, <mask>
<mask_test> := {INSIDE, OUTSIDE}
<old_mask_test> := {ALLI, ANYO}
<mask> := {TA, TB, TC. TD}

Note: Old mask test keywords ALLI and ANYO imply testing of <trace> against the mask waveform TD.

Response Format

<trace>:PArameter_VAlue <parameter>,<value>,<state>[,... ,<parameter>,<value>,<state>]

<value> := decimal numeric value
<state> := {OK, AV, PT, IV, NP, GT, LT, OF, UF, OU}

Note: If <parameter> is not specified, or equal to ALL, all the parameters followed by their values and states are returned.

EXAMPLE (GPIB)

The following query reads the risetime of Trace B (TB):

CMD$="TB:PAVA? RISE": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD (SCOPE%,RD$): PRINT RD$
Response message:
TB:PAVA RISE,3.6E–9S,OK

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PARAMETER_STATISTICS?

* 4–channel oscilloscopes only

MISCELLANEOUS

PASS_FAIL_CONDITION, PFCO

Command/Query

DESCRIPTION

The PASS_FAIL_CONDITION command adds a Pass/Fail test condition or a custom parameter at the specified line on the Pass/Fail or Custom Parameter display.

The PASS_FAIL_CONDITION? query indicates the current Pass/Fail test setup or the current selection of custom parameters at the specified line.

Note: Up to five test conditions (or custom parameters) can be specified at five different display lines on the screen. The PASS_FAIL_CONDITION command deals with one line at a time.

Notation			
GT	Greater than	LT	Lower than

COMMAND SYNTAX

Pass_Fail_COndition [<line>,<trace>,<param>[,<rel_op>[,<ref_value>]]]

<line> := {1,2,3,4,5}
<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}
<param> := parameters listed for the PARAMETER_VALUE?, PAVA? query, with the exception of ALL
<rel_op> := {GT,LT}
<ref_value> := –1e + 15 to 1e15

Note: The PFCO command with no arguments (i.e. "PFCO") deletes all conditions. The PFCO command with a single argument (i.e. "PFCO <line>") deletes the condition at <line>.

Note: Old keywords ALLI and ANYO imply a test of <trace> against the mask waveform TD.

Alternative form of command for mask tests:

Pass_Fail_COndition [<line>,<trace>,<mask_test>,<mask>]
<mask_test> := {INSIDE, OUTSIDE}
<mask> := {TA, TB, TC. TD}

* 4–channel oscilloscopes only

QUERY SYNTAX

PFCO? <line>

Response format

PFCO <line>,<trace>,<param>,<rel_op>,<ref_value>

Alternative form of response for mask tests:

PFCO <line>,<trace>,<mask_test>,<mask>

EXAMPLE (GPIB)

The following command sets the first test condition in the list to be "frequency on Channel 1 lower than 10 kHz":

```
CMD$ = "PFCO 1,C1,FREQ,LT,10000":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PASS_FAIL_COUNTER, PASS_FAIL_DO, PASS_FAIL_MASK, PARAMETER_VALUE

MISCELLANEOUS

PASS_FAIL_COUNTER, PFCT

Command/Query

DESCRIPTION

The PASS_FAIL_COUNTER command resets the Passed/Failed acquisitions counters. The PASS_FAIL_COUNTER? query returns the current counts.

COMMAND SYNTAX

Pass_Fail_CounTer

QUERY SYNTAX

Pass_Fail_CounTer?

Response Format

Pass_Fail_CounTer <PASS>,<value>,OF,<value>

<value> := 0 to 999999

EXAMPLE (GPIB)

The following query reads the counters:

```
CMD$="PFCT?";
CALL IBWRT(SCOPE%,CMD$)
```

Response message:
PFCT PASS, 8, OF, 9

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PASS_FAIL_DO, PASS_FAIL_MASK, PARAMETER_VALUE

MISCELLANEOUS

PASS_FAIL_DO, PFDO

Command/Query

DESCRIPTION

The PASS_FAIL_DO command defines the desired outcome and the actions that have to be performed by the oscilloscope after a Pass/Fail test. The PASS_FAIL_DO? query indicates which actions are currently selected.

Notation	
BEEP	Emit a beep
PULS	Emit a pulse on the CAL connector
SCDP	Make a hardcopy
STO	Store in Memory or on storage media
STOP	Stop acquisition

COMMAND SYNTAX

Pass_Fail_DO [<outcome>[,<act>[,<act>...]]]

<outcome> := {PASS,FAIL}

<act> := {STOP, SCDP, STO}

Note: The BEEP command is accepted only on models equipped with the CLBZ hardware option.

Note: The PULS command is accepted only on models equipped with the CKIO software option.

Note: The PFDO command with no arguments (i.e. "PFDO") deletes all actions.

Note: The STO command performs the store operation as described in the Waveform Store chapter in the User's Manual.

Note: After every pass or fail detected, the instrument sets the INR bit 12.

QUERY SYNTAX

Pass_Fail_DO?

Response Format

Pass_Fail_DO [<pass_fail>[,<act>[,<act>...]]]

EXAMPLE (GPIB)

The following command forces the oscilloscope to stop acquiring when the test passes:

```
CMD$="DO PASS,STOP":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS PASS_FAIL_COUNTER, PASS_FAIL_MASK, PARAMETER_VALUE?, CURSOR_MEASURE, CURSOR_SET, INR?, BUZZER

MISCELLANEOUS

PASS_FAIL_MASK, PFMS

Command

DESCRIPTION

The PASS_FAIL_MASK command generates a tolerance mask around a chosen trace and stores the mask into the selected memory.

COMMAND SYNTAX

Pass_Fail_MaSk [<trace>[,<htol>[,<vtol>[,<mask>]]]]
<trace> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}
<htol> := 0.0 to 5.0
<vtol> := 0.0 to 4.0
<mask> := {M1, M2, M3, M4}

Note: if any arguments are missing, the previous settings will be used.

The alternative form of command:

Pass_Fail_MaSk INVT [,<mask>]

inverts the mask in the selected mask memory. If <mask> is missing, M4 is implied.

EXAMPLE (GPIB)

The following command generates a tolerance mask around the Channel 1 trace and stores it in M2:

```
CMD$ = "PASS_FAIL_MASK C1,0.2,0.3,M2":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO, PARAMETER_VALUE?

* 4–channel oscilloscopes only

DISPLAY

PERSIST, PERS

Command/Query

DESCRIPTION

The PERSIST command enables or disables the persistence display mode.

COMMAND SYNTAX

PERSist <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

PERSist?

Response format

PERSist <mode>

EXAMPLE (GPIB)

The following code turns the persistence display ON.

```
CMD$="PERS ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

XY_DISPLAY

CURSOR

PER_CURSOR_SET, PECS

Command/Query

DESCRIPTION

The PER_CURSOR_SET command allows the user to position any one of the six independent cursors at a given screen location. The position of the cursor can be modified or queried even if the cursor is not currently displayed on the screen.

The PER_CURSOR_SET? query indicates the current position of the cursor(s).

The vertical cursor positions are the same as those controlled by the CURSOR_SET command

Notation			
HABS	Horizontal absolute	VABS	Vertical absolute
HDIF	Horizontal difference	VDIF	Vertical difference
HREF	Horizontal reference	VREF	Vertical reference

COMMAND SYNTAX

<trace>: PEr_Cursor_Set <cursor>,<position>[,<cursor>, <position>,...,<cursor>,<position>

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}

<cursor> := {HABS, HDIF, HREF, VABS, VDIF, VREF}

<position> := 0 to 10 DIV (horizontal)
–29.5 to 29.5 DIV (vertical)

Note 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be in any order and may be restricted to those variables to be changed.

QUERY SYNTAX

<trace>: PEr_Cursor_Set? <cursor>[<cursor,...,<cursor>]

<cursor> := {HABS, HDIF, HREF, VABS, VDIF, VREF, ALL}

Note: If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.

Response Format

PEr_Cursor_Set <cursor>,<position>[,<cursor>,<position>,..., <cursor>,<position>

* 4–channel oscilloscopes only

EXAMPLE (GPIB)

The following code positions the HREF and HDIF cursors at +2.6 DIV and +7.4 DIV respectively, using Channel 2 as a reference.

```
CMD$="C2:PECS HREF,2.6 DIV,HDIF,7.4DIV"
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PERSIST, PER_CURSOR_VALUE,

CURSOR

PER_CURSOR_VALUE?, PECV?

Query

DESCRIPTION

The PER_CURSOR_VALUE? query returns the values measured by the specified cursors for a given trace.

Notation			
HABS	Horizontal absolute	VABS	Vertical absolute
HREL	Horizontal relative	VREL	Vertical relative

QUERY SYNTAX

<trace>:PEr_Cursor_Value? <cursor>[,<cursor>,...,<cursor>]

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*}
<cursor> := {HABS, HREL, VABS, VREL, ALL}

Note: If <cursor> is not specified, ALL will be assumed.

Response format

<trace>:PEr_Cursor_Value <cursor>,<value>[,<cursor>,<value>,...,<cursor>,<value>]

EXAMPLE (GPIB)

The following code returns the value measured with the vertical relative cursor on Channel 1.

```
CMD$="C1:PECV? VREL": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message

```
C1:PECV VREL,56 MV
```

RELATED COMMANDS

CURSOR_MEASURE, PERSIST, PER_CURSOR_SET

* 4–channel oscilloscopes only

STATUS

*PRE

Command/Query

DESCRIPTION

The *PRE command sets the PaRallel poll Enable register (PRE). The lowest 8 bits of the Parallel Poll Register (PPR) are composed of the STB bits. The *PRE command allows the user to specify which bit(s) of the parallel poll register will affect the 'ist' individual status bit.

The *PRE? query reads the contents of the PRE register. The response is a decimal number which corresponds to the binary sum of the register bits.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

PRE <value>

<value> := 0 to 65 535

QUERY SYNTAX

*PRE?

Response format

*PRE <value>

EXAMPLE (GPIB)

The following command will cause the 'ist' status bit to become 1 as soon as the MAV bit (bit 4 of STB, i.e. decimal 16) is set. This yields the PRE value 16.

CMD$="*PRE 16": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

*IST?

SAVE/RECALL SETUP

*RCL
Command

DESCRIPTION

The *RCL command sets the state of the instrument using one of the five non–volatile panel setups by recalling the complete front–panel setup of the instrument. Panel setup 0 corresponds to the default panel setup.

The *RCL command produces the opposite effect of the *SAV command.

If the desired panel setup is not acceptable, the Execution error status Register (EXR) is set and the EXE bit of the standard Event Status Register (ESR) is set.

COMMAND SYNTAX

*RCL <panel_setup>

<panel_setup> := 0 to 4

EXAMPLE (GPIB)

The following code recalls the instrument setup previously stored in panel setup 3:

```
CMD$="*RCL 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *SAV, EXR?

WAVEFORM TRANSFER

RECALL, REC

Command

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The RECALL command recalls a waveform file from mass storage into one of the internal memories M1 to M4.

COMMAND SYNTAX

<memory>:RECall DISK, <dev>, FILE,'<filename>'

<memory> := {M1, M2, M3, M4}

<dev> ;= {CARD, FLPY}

<filename> := an alphanumeric string of up to 8 characters, followed by a dot and an extension of up to 3 digits.

EXAMPLE (GPIB)

The following command recalls a waveform file called "SC1.001" from the memory card into Memory M1:

```
CMD$="M1:REC DISK,CARD,FILE,'SC1.001' ":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

FUNCTION_STATE, STORE, INR?

SAVE/RECALL SETUP

RECALL_PANEL, RCPN

Command

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The RECALL_PANEL command recalls a front–panel setup from mass storage.

COMMAND SYNTAX

ReCall_PaNel DISK, <dev>, FILE,'<filename>'

<dev> := {CARD, FLPY}

<filename> := a string of up to 8 characters. Any extension will be ignored. Front–panel setup files are always stored with the extension ".PNL" on the card.

EXAMPLE (GPIB)

The following command recalls the front–panel setup from file P012.PNL on the floppy disk:

```
CMD$="RCPN DISK, FLPY, FILE,'P012'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *SAV, STORE_PANEL, *RCL

SAVE/RECALL SETUP

*RST

Command

DESCRIPTION

The *RST command initiates a device reset. The *RST sets all 8 traces to the GND line, recalls the default setup and causes a calibration to be performed.

COMMAND SYNTAX

*RST

EXAMPLE (GPIB)

This example resets the oscilloscope:

```
CMD$="*RST": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*CAL?, *RCL

ACQUISITION

SAMPLE_CLOCK, SCLK

Command/Query

Oscilloscopes fitted with the CKIO option

DESCRIPTION

The SAMPLE_CLOCK command allows the user to control the use of an external time base. The user sets the number of data points that will be acquired when the oscilloscope is using the external clock.

COMMAND SYNTAX

Sample_CLocK <state>[,<recordlength>]

<state> := {INT, ECL, LV0, TTL}
<recordlength> := {50, 100, 200, 500, 1K, 2K, 5K, 10K, 20K, 50K, 100K, 200K, 500K, 1M }

Note: The record length cannot be chosen larger than the maximum available memory of the model being used. (The parameter <recordlength> is initially set to 10K).

QUERY SYNTAX

Sample_CLocK?

Response Format

Sample_CLocK <state>,<recordlength>

EXAMPLE

The following command sets the oscilloscope to use the external clock with 1000 data point records.

CMD$ = "SCLK EXT,1000": CALL IBWRT(SCOPE%,CMD$)

SAVE/RECALL SETUP

*SAV

Command

DESCRIPTION

The *SAV command stores the current state of the instrument in non–volatile internal memory. The *SAV command stores the complete front–panel setup of the instrument at the time the command is issued.

*Note: The communication parameters (those modified by commands COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER and WAVEFORM_SETUP) and the enable registers associated with the status reporting system (*SRE, *PRE, *ESE, INE) are not saved by this command.*

COMMAND SYNTAX

*SAV <panel_setup>

<panel_setup> := 1 to 4

EXAMPLE (GPIB)

The following code saves the current instrument setup in panel setup 3:

```
CMD$="*SAV 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *RCL

HARD COPY

SCREEN_DUMP, SCDP

Command/Query

DESCRIPTION

The SCREEN_DUMP command causes the oscilloscope to dump the screen contents onto the hard copy device. This command will halt oscilloscope activities.

The time/date stamp which appears on the print corresponds to the time at which the command was executed.

COMMAND SYNTAX

SCreen_DumP

QUERY SYNTAX

SCreen_DumP?

Response format

SCreen_DumP <status>
<status> := {OFF}

EXAMPLE (GPIB)

The following code initiates a screen dump:

CMD$="SCDP"; CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

INR?, HARDCOPY_SETUP, HARDCOPY_TRANSMIT

ACQUISITION

SEGMENTS, SEGS

Command/Query

DESCRIPTION

The SEGMENTS command sets the number of segments for sequence mode acquisition.

The response to the SEGMENTS? query indicates the number of segments which is set in the oscilloscope.

Note: The SEGMENTS command is only to be used for compatibility with the 9400 series oscilloscopes. In all other cases, the SEQUENCE command should be used.

COMMAND SYNTAX

SEGmentS <segments>

<segments> := 2 to maximum

See the SEQUENCE command for the model–dependent maximum value.

QUERY SYNTAX

SEGmentS?

Response Format

SEGmentS <segments>

EXAMPLE

The following command sets the segment count to 100:

```
CMD$ = "SEGS 100": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_MODE, SEQUENCE

DISPLAY

SELECT, SEL

Command/Query

DESCRIPTION

The SELECT command selects the specified trace for manual display control. An environment error (see table on page 5–49) is generated if the specified trace is not displayed.

The SELECT? query returns the selection status of the specified trace.

COMMAND SYNTAX

<trace>:SELect

<trace> := {TA, TB, TC, TD}

QUERY SYNTAX

<trace>:SELect?

Response format

<trace>:SELect <mode>

<mode> := {ON, OFF}

EXAMPLE (GPIB)

The following command selects Trace B (TB):

CMD$="TB:SEL": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

TRACE

ACQUISITION

SEQUENCE, SEQ

Command/Query

DESCRIPTION

The SEQUENCE command sets the conditions for the sequence mode acquisition.

The response to the SEQUENCE? query gives the conditions for the sequence mode acquisition.

COMMAND SYNTAX

SEQuence mode [,<segments>[,<max_size>]]

	L models	M models	Others
<segments> :=	2 to 2000	2 to 200	2 to 50
<max_size> :=	50 100 250 500 1000 2500 5K 10K 25K 50K	50 100 250 500 1000 2500 5K 10K 25K	50 100 250 500 1000 2500 5K

Note: <max_size> is a keyword

<mode> := {OFF, ON, WRAP}

QUERY SYNTAX

SEQuence?

Response format

SEQuence <mode> , <segments> , <max_size>

EXAMPLE

The following command sets the segment count to 43, maximum segment size to 250 samples, and turns the sequence mode ON:

CMD$="SEQ ON,43,250": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

TRIG_MODE

STATUS

*SRE

Command/Query

DESCRIPTION

The *SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request. Refer to the table on page 5–107 for an overview of the available summary messages.

A summary message bit is enabled by writing a 1 into the corresponding bit location. Conversely, writing a 0 into a given bit location prevents the associated event from generating a service request (SRQ). Clearing the SRE register disables SRQ interrupts.

The *SRE? query returns a value which when converted to a binary number represents the bit settings of the SRE register. Note that bit 6 (MSS) cannot be set and its returned value is always zero.

Note: This command can be executed in both local and remote modes.

COMMAND SYNTAX

*SRE <value>

<value> := 0 to 255

QUERY SYNTAX

*SRE?

Response format

*SRE <value>

EXAMPLE (GPIB)

The following command allows an SRQ to be generated as soon as the MAV summary bit (bit 4, i.e. decimal 16) and/or the INB summary bit (bit 0, i.e. decimal 1) in the STB register are set. Summing these two values yields the SRE mask 16 + 1 = 17.

```
CMD$="*SRE 17": CALL IBWRT(SCOPE%,CMD$)
```

STATUS

***STB?**

Query

DESCRIPTION

The *STB? query reads the contents of the 488.1 defined status register (STB), and the Master Summary Status (MSS). The response represents the values of bits 0 to 5 and 7 of the Status Byte register and the MSS summary message.

The response to a *STB? query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message. Refer to the table on page 5–107 for further details on the status register structure.

QUERY SYNTAX

*STB?

Response format

*STB <value>

<value> := 0 to 255

EXAMPLE (GPIB)

The following instruction reads the status byte register:

```
CMD$="*STB?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
*STB 0
```

RELATED COMMANDS

ALL_STATUS?, *CLS, *PRE, *SRE

Bit	Value	Name		Description	Note
7	128	DI07	0	Reserved for future use	
6	64	MSS/RQS			(1)
		MSS =1		at least 1 bit in STB masked by SRE is 1	(2)
		RQS =1		service is requested	
5	32	ESB	1	an ESR enabled event has occurred	(3)
4	16	MAV	1	Output queue is not empty	(4)
3	8	DIO3	0	Reserved	
2	4	VAB	1	a command data value has been adapted	(5)
1	2	DIO1	0	Reserved	
0	1	INB	1	an enabled INternal state change has occurred.	(6)

Status Byte Register (STB)

Notes:

(1) The Master Summary Status (MSS) indicates that the instrument requests service while the Service Request status – when set – specifies that the oscilloscope issued a service request. Bit position 6 depends on the polling method:

Bit 6 = MSS if a *STB? query is received
= RQS if serial polling is conducted

(2) Example: If SRE = 10 and STB = 10 then MSS = 1. If SRE = 010 and STB = 100 then MSS=0.

(3) The Event Status Bit (ESB) indicates whether or not one or more of the enabled IEEE 488.2 events have occurred since the last reading or clearing of the Standard Event Status Register (ESR). ESB is set if an enabled event becomes true (1).

(4) The Message AVailable bit (MAV) indicates whether or not the Output queue is empty. The MAV summary bit is set true (1) whenever a data byte resides in the Output queue.

(5) The Value Adapted Bit (VAB) is set true (1) whenever a data value in a command has been adapted to the nearest legal value. For instance, the VAB bit would be set if the time base is redefined as 2.5 µsec/div since the adapted value is 2 µsec/div.

(6) The INternal state Bit (INB) is set true (1) whenever certain enabled internal states are entered. For further information, refer to the INR? query.

ACQUISITION

STOP

Command

DESCRIPTION

The STOP command immediately stops the acquisition of a signal. If the trigger mode is AUTO or NORM it will change to trigger mode STOPPED to prevent further acquisition.

COMMAND SYNTAX

STOP

EXAMPLE

The following command stops the acquisition process:

```
CMD$ = "STOP": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

ARM_ACQUISITION, TRIG_MODE, WAIT

WAVEFORM TRANSFER

STORE, STO

Command

DESCRIPTION

The STORE command stores the contents of the specified trace into one of the internal memories M1 to M4 or to mass storage.

COMMAND SYNTAX

STOre [<trace>, <dest>]
<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*, ALL_DISPLAYED}
<dest> := {M1, M2, M3, M4, CARD, FLPY}

Note: If the STORE command is sent without any argument, all traces currently enabled in the Store Setup will be stored. This setup can be modified using the STORE_SETUP command.

EXAMPLE (GPIB)

The following command stores the contents of Trace A (TA) into Memory 1 (M1):

CMD$=STO TA,M1": CALL IBWRT(SCOPE%,CMD$)

The following command stores all currently displayed waveforms on the memory card:

CMD$="STO ALL_DISPLAYED, CARD'":
CALL IBWRT(SCOPE%,CMD$)

The following command executes the storage operation which is currently defined in the Storage Setup (see command STORE_SETUP):

CMD$="STO": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

STORE_SETUP, RECALL

* 4–channel oscilloscopes only

SAVE/RECALL PANEL

STORE_PANEL, STPN

Command

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The STORE_PANEL command stores the complete front–panel setup of the instrument, at the time the command is issued, into a file on mass storage.

*Note: The communication parameters (those modified by commands COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER and WAVEFORM_SETUP) and the enable registers associated with the status reporting system (*SRE, *PRE, *ESE, INE) are not saved by this command.*

COMMAND SYNTAX

STore_PaNel DISK, <dev>, FILE,'<filename>'

<dev> := {CARD, FLPY}

<filename> := a string of up to 8 characters. Any extension will be ignored. Front–panel setup files are always stored with the extension ".PNL" on the card.

Note: If no filename (or an empty string) is supplied, the oscilloscope generates a filename according to its internal rules.

EXAMPLE (GPIB)

The following code saves the current instrument setup to the memory card in a file called "DIODE.PNL":

```
CMD$="STPN DISK, CARD, FILE,'DIODE'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PNSU, *SAV, RECALL_PANEL, *RCL

WAVEFORM TRANSFER

STORE_SETUP, STST

Command/Query

DESCRIPTION

The STORE_SETUP command controls the way in which traces will be stored. A single trace or all displayed traces may be enabled for storage. This applies to autostoring or to the STORE, STO command. Traces may be autostored to mass storage after each acquisition until the card or floppy becomes full (FILL), or forever (WRAP), replacing the oldest traces by new ones.

The STORE_SETUP? query returns the current mode of operation of Autostore, the current trace selection, and the current destination.

COMMAND SYNTAX

STore_SeTup [<trace>,<dest>] [,AUTO,<mode>]

<trace> := {TA, TB, TC, TD, C1, C2, C3*, C4*, ALL_DISPLAYED}

<dest> := {M1, M2, M3, M4, CARD, FLPY}

<mode> := {OFF, WRAP, FILL}

QUERY SYNTAX

STore_SeTup?

Response format

STore_SeTup <trace>,<dest>,AUTO,<mode>

EXAMPLE (GPIB)

The following command selects Channel 1 to be stored. It enables an "autostore" to the card until no more space is left on the memory card (AUTO, FILL).

```
CMD$="STST C1, CARD, AUTO,FILL":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STORE, INR?

* 4–channel oscilloscopes only

WAVEFORM TRANSFER

STORE_TEMPLATE, STTM

Command

Oscilloscopes fitted with the MC01 or FD01 options

DESCRIPTION

The STORE_TEMPLATE command stores the instrument's waveform template on a memory card or floppy disk. A filename is automatically generated in the form of "LECROYvv.TPL" where "vv" is the 2–digit revision number.

Note: As an example, the file name generated will be LECROY21.TPL for revision 2.1.

Refer to Section 6 for further information about the waveform template.

COMMAND SYNTAX

STore_TeMplate DISK, <dev>

<dev> := {CARD, FLPY}

EXAMPLE (GPIB)

The following code stores the current waveform template on the memory card for future reference:

```
CMD$="STTM DISK, CARD": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TEMPLATE?

WAVEFORM TRANSFER

TEMPLATE?, TMPL?

Query

DESCRIPTION

The TEMPLATE? query produces a copy of the template which formally describes the various logical entities making up a complete waveform. In particular, the template describes in full detail the variables contained in the descriptor part of a waveform. Refer to chapter 6 for further information.

QUERY SYNTAX

TeMPLate?

Response format

TeMPLate "<template>"

<template> := A variable length string detailing the structure of a waveform.

RELATED COMMANDS

INSPECT?

ACQUISITION

TIME_DIV, TDIV

Command/Query

DESCRIPTION

The TIME_DIV command modifies the time–base setting. The new time–base setting may be specified with suffixes NS for nanoseconds, US for microseconds, MS for milliseconds, S for seconds or KS for kiloseconds. An out–of–range value causes the VAB bit (bit 2) in the STB register (see table on page 5–107) to be set.

The oscilloscope will force random interleaved sampling (RIS) for time–base settings in the "ON" only range, and single–shot sampling for time–base settings in the "OFF" only range. For time–base settings in the "Selectable" range, the INTERLEAVED command allows the user to choose the required sampling mode.

RIS	10K	50K
"ON" only	1 ns/div to 20 ns/div	1 ns/div to 20 ns/div
Selectable "ON"/"OFF"	50 ns/div to 5 µs/div	50 ns/div to 20 µs/div
"OFF" only	10 µs/div to 1000 s/div	50 µs/div to 1000 s/div

Sequence mode acquisitions also force single–shot sampling and therefore cannot be performed for time–base settings in the RIS "ON" only range (see table above).

The TIME_DIV? query returns the current time–base setting.

COMMAND SYNTAX

Time_DIV <value>

<value> := 1 NS to 1 KS

Note: The suffix S (seconds) is optional.

QUERY SYNTAX

Time_DIV?

Response Format

Time_DIV <value>

EXAMPLE (GPIB)

The following command sets the time base to 500 µsec/div:

```
CMD$="TDIV 500US": CALL IBWRT(SCOPE%,CMD$)
```

The following command sets the time base to 2 msec/div:

```
CMD$="TDIV 0.002": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INTERLEAVED, TRIG_DELAY, TRIG_MODE

DISPLAY

TRACE, TRA

Command/Query

DESCRIPTION

The TRACE command enables or disables the display of a trace. An environment error (see table on page 5–49) is set if an attempt is made to display more than four waveforms.

The TRACE? query indicates whether the specified trace is displayed or not.

COMMAND SYNTAX

<trace>:TRAce <mode>

<trace> := {C1, C2, C3*, C4*, TA, TB, TC, TD}
<mode> := {ON, OFF}

QUERY SYNTAX

<trace>:TRAce?

Response format

<trace>:TRAce <mode>

EXAMPLE (GPIB)

The following command displays Trace A (TA):

CMD$="TA:TRA ON": CALL IBWRT(SCOPE%,CMD$)

* 4–channel oscilloscopes only

ACQUISITION

*TRG

Command

DESCRIPTION

The *TRG command executes an ARM command.

Note: The *TRG command is the equivalent of the 488.1 GET (Group Execute Trigger) message.

COMMAND SYNTAX

*TRG

EXAMPLE (GPIB)

The following command enables signal acquisition:

CMD$="*TRG": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

ARM_ACQUISITION, STOP, WAIT

ACQUISITION

TRIG_COUPLING, TRCP

Command/Query

DESCRIPTION

The TRIG_COUPLING command sets the coupling mode of the specified trigger source. The trigger slope is automatically changed to positive when the trigger coupling is set to HFDIV.

Note: HFDIV is indicated as HF in the trigger setup menus.

The TRIG_COUPLING? query returns the trigger coupling of the selected source.

COMMAND SYNTAX

<trig_source>:TRig_CouPling <trig_coupling>

<trig_source> := {C1, C2, C3*, C4*, EX, EX10}

<trig_coupling> := {AC, DC, HFREJ, LFREJ, HFDIV}}

QUERY SYNTAX

<trig_source>:TRig_CouPling?

Response format

<trig_source>:TRig_CouPling <trig_coupling>

EXAMPLE (GPIB)

The following command sets the coupling mode of the trigger source Channel 2 to high frequency reject:

CMD$="C2:TRCP HFREJ": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

TRIG_SELECT

* 4–channel oscilloscopes only

ACQUISITION

TRIG_DELAY, TRDL

Command/Query

DESCRIPTION

The TRIG_DELAY command sets the time at which the trigger is to occur with respect to the first acquired data point (displayed at the left–hand edge of the screen).

The command expects positive trigger delays to be expressed as a percentage of the full horizontal screen (this mode is called pre–trigger acquisition as data are acquired before the trigger occurs). Negative trigger delays must be given in seconds (this mode is called post–trigger acquisition as the data are acquired after the trigger has occurred).

If a value outside the range –10 000 div × time/div and 100% is specified, the trigger time will be set to the nearest limit and the VAB bit (bit 2) will be set in the STB register.

The response to the TRIG_DELAY? query indicates the trigger time with respect to the first acquired data point. Positive times are expressed as a percentage of the full horizontal screen and negative times in seconds.

COMMAND SYNTAX

TRig_DeLay <value>

<value> := 0.00 PCT to 100.00 PCT (pretrigger)
–20 PS to –10 MAS (post–trigger)

Note: The suffix is optional. For positive numbers the suffix PCT is assumed. For negative numbers the suffix S is assumed. MAS is the suffix for Msec (megaseconds), useful only for extremely large delays at very slow time bases.

QUERY SYNTAX

TRig_DeLay?

Response format

TRig_DeLay <value>

EXAMPLE (GPIB)

The following command sets the trigger delay to – 20 sec (post–trigger):

```
CMD$="TRDL –20S": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TIME_DIV

ACQUISITION

TRIG_LEVEL, TRLV

Command/Query

DESCRIPTION

The TRIG_LEVEL command adjusts the trigger level of the specified trigger source. An out–of–range value will be adjusted to the closest legal value and will cause the VAB bit (bit 2) in the STB register (see table on page 5–107) to be set.

The range of values is as follows:

- ± 5 times the total V/div setting – with a channel as the trigger source.
- ± 2 V with EXT as trigger source
- ± 20 V with EXT/10 as trigger source.

The TRIG_LEVEL? query returns the current trigger level.

COMMAND SYNTAX

<trig_source>:TRig_LeVel <trig_level>

<trig_source> := {C1, C2, C3*, C4*, EX, EX10}

<trig_level> := –25V to 25V (maximum range)

Note: The suffix V is optional.

QUERY SYNTAX

<trig_source>:TRig_LeVel?

Response format

<trig_source>:TRig_LeVel <trig_level>

EXAMPLE (GPIB)

The following command adjusts the trigger level of Channel 2 to –3.4 V:

```
CMD$="C2:TRLV –3.4V": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_SELECT

* 4–channel oscilloscopes only

ACQUISITION

TRIG_MODE, TRMD

Command/Query

DESCRIPTION

The TRIG_MODE command specifies the trigger mode.
The TRIG_MODE? query returns the current trigger mode.

COMMAND SYNTAX

TRig_MoDe <mode>
<mode> := {AUTO, NORM, SINGLE, STOP}

QUERY SYNTAX

TRig_MoDe?

Response format

TRig_MoDe <mode>

EXAMPLE (GPIB)

The following command selects the normal mode:
CMD$="TRMD NORMAL": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

ARM_ACQUISITION, STOP, TRIG_SELECT, SEQUENCE

ACQUISITION

TRIG_SELECT, TRSE

Command/Query

DESCRIPTION

The TRIG_SELECT command selects the condition that will trigger the acquisition of waveforms. Depending on the trigger type, additional parameters have to be specified.

The additional parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be given in any order and may be restricted to those variables to be changed.

The TRIG_SELECT? query returns the current trigger condition.

Trigger Notation

DROP	Dropout	EDGE	Edge
EV	Event	GLIT	Glitch
HT	Hold type	HV	Hold value
HV2	Second hold value	IL	Interval larger
INTV	Interval	IS	Interval smaller
I2	Interval–width window	OFF	No hold–off on wait
PL	Pulse larger	PS	Pulse smaller
P2	Pulse–width window	QL	Qualifier
SNG	Single source	SQ	State qualified
SR	Source	STD	Standard
TEQ	Time/event qualified	TI	Time
TL	Time within		

TV Trigger Notation

CHAR	Characteristics	FLD	Field
FLDC	Field count	ILAC	Interlace
LINE	Line	LPIC	Lines per picture

HT and HV do not apply to the standard trigger.

NON TV TRIGGER COMMAND SYNTAX

TRig_SElect <trig_type>,SR,<source>,QL,<source>,
HT,<hold_type>,HV,<hold_value>,HV2,<hold value>

<trig_type> := {STD, SNG, SQ, TEQ, DROP, EDGE, GLIT, INTV, TV}

<source> := {C1, C2, C3*, C4*, LINE, EX, EX10}

<hold_type> := {TI, TL, EV, PS, PL, IS, IL, P2, I2, OFF}

<hold_value> := 25NS to 20S for TI
1 to 99,999,999 (for EV)
2.5NS to 20S for PS and PL
10NS to 20S for IS and IL

Note: The suffix S (seconds) is optional.

QUERY SYNTAX

TRig_SElect?

Response format

TRig_SElect <trig_type>,SR,<source>,HT,<hold_type>, HV,<hold_value>

EXAMPLE (GPIB)

The following command selects the single–source trigger with Channel 1 as trigger source. Hold type and hold value are chosen as "Pulse smaller" than 20 µsec

```
CMD$="TRSE SNG,SR,C1,HT,PS,HV,20 US"
CALL IBWRT(SCOPE%,CMD$)
```

TV TRIGGER COMMAND SYNTAX

TRig_SElect TV,SR,<source>,FLDC,<field_count>,FLD,<field>, CHAR,<characteristics>,LPIC@,<lpic>,ILAC@,<ilace>, LINE,<line>

<source> := {C1, C2, C3*, C4*, LINE, EX, EX10}
<field_count> := {1, 2, 4, 8}
<field> := 1 to field_count
<characteristics> := { NTSC,PALSEC,CUST50,CUST60}
<lpic> := 1 to 1500
<ilace> := {1, 2, 4, 8}
<line> := 1 to 1500

Note: The FLD value is interpreted with the current FLDC value. The LINE value is interpreted with the current FLD and CHAR values.

QUERY SYNTAX

TRig_SElect?

Response format

TRig_SElect TV,SR,<source>,FLDC,<field_count>,FLD,<field>, CHAR,<characteristic>,LINE,<line>

* 4–channel oscilloscopes only @ for CUST50 and CUST60 only

EXAMPLE (GPIB)

CMD$ = "TRSE TV,SR,EX,FLDC,8,FLD,3,CHAR,
PALSEC,LINE,17"

RELATED COMMANDS

TRIG_COUPLING, TRIG_LEVEL, TRIG_MODE, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_SLOPE, TRSL

Command/Query

DESCRIPTION

The TRIG_SLOPE command sets the trigger slope of the specified trigger source. An environment error (see table on page 5–49) will be generated when an incompatible TRSL order is received while the trigger coupling is set to HFDIV (see TRIG_COUPLING).

The TRIG_SLOPE? query returns the trigger slope of the selected source.

COMMAND SYNTAX

<trig_source>:TRig_SLope <trig_slope>

<trig_source> := {C1, C2, C3*, C4*, LINE, EX, EX10}

<trig_slope> := {NEG, POS, WINDOW}

QUERY SYNTAX

<trig_source>:TRig_SLope?

Response format

<trig_source>:TRig_SLope <trig_slope>

EXAMPLE (GPIB)

The following command sets the trigger slope of Channel 2 to negative:

```
CMD$="C2:TRSL NEG": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_SELECT, TRIG_WINDOW

* 4–channel oscilloscopes only

ACQUISITION

TRIG_WINDOW, TRWI

Command/Query

DESCRIPTION

The TRIG_WINDOW command sets the window amplitude in volts on the current Edge–trigger source. The window is centered around the Edge–trigger level.

The TRIG_WINDOW? query returns the current window amplitude.

COMMAND SYNTAX

TRig_WIndow <value>

<value> := 0 to 25 V (maximum range)

Note: The suffix V is optional.

QUERY SYNTAX

TRig_WIndow?

Response format

TRig_WIndow <trig_level>

EXAMPLE

The following command adjusts the window size to +0.5 V:

```
CMD$="TRWI 0.5V": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_LEVEL, TRIG_SLOPE

MISCELLANEOUS

*TST?

Query

DESCRIPTION

The *TST? query performs an internal self–test. The response indicates if the self–test detected any errors. The self–test includes testing the hardware of all channels, the time base and the trigger circuits.

Hardware failures are identified by a unique binary code in the returned <status> number (see table on page 5–12). A "0" response indicates that no failures occurred.

Note: This query is only accepted in remote mode.

QUERY SYNTAX

*TST?

Response Format

*TST <status>

<status>:= 0 self–test successful

EXAMPLE (GPIB)

This example causes a self–test to be performed:

```
CMD$="*TST?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure):

```
*TST 0
```

RELATED COMMANDS

*CAL?

STATUS

URR?

Query

DESCRIPTION

The URR? query reads and clears the contents of the User Request status Register (URR). The URR register specifies which button in the menu field was pressed.

In the remote mode, the URR register indicates the last button which was pressed, provided that it was activated with a KEY command. In local mode, the URR register indicates whether the CALL HOST button has been pressed. If no menu button has been pressed since the last URR? query, the value 0 is returned. The figure on page 5–70 shows the button assignments on the instrument.

QUERY SYNTAX

URR?

Response format

URR <value>

<value> := 0 to 6, 100

EXAMPLE (GPIB)

The following instruction reads the contents of the URR register:

```
CMD$="URR?": CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:
URR 0

RELATED COMMANDS

CALL_HOST, KEY, ALL_STATUS?, *CLS

Value	Description
0	no button has been pressed
2	button 2 has been pressed.
3	button 3 has been pressed.
4	button 4 has been pressed.
5	button 5 has been pressed.
6	button 6 has been pressed.
100	The "Call Host" key (button 7 in root menu) has been pressed.

User Request Status Register Structure (URR)

DISPLAY

VERT_MAGNIFY, VMAG

Command/Query

DESCRIPTION

The VERT_MAGNIFY command vertically expands the specified trace. The command is executed even if the trace is not displayed.

The maximum magnification allowed depends on the number of significant bits associated with the data of the trace.

The VERT_MAGNIFY? query returns the magnification factor of the specified trace.

COMMAND SYNTAX

<trace>:Vert_MAGnify <factor>

<trace> := {TA, TB, TC, TD}
<factor> := 4×10^{-3} to 50 (maximum)

QUERY SYNTAX

<trace>:Vert_MAGnify?

Response format

<trace>:Vert_MAGnify <factor>

EXAMPLE (GPIB)

The following command enlarges the vertical amplitude of Trace A by a factor of 3.45 with respect to its original amplitude:

```
CMD$="TA:VMAG 3.45": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

VERT_POSITION

DISPLAY

VERT_POSITION, VPOS

Command/Query

DESCRIPTION

The VERT_POSITION command adjusts the vertical position of the specified trace on the screen. The VERT_POSITION command does not affect the original offset value obtained at acquisition time.

The VERT_POSITION? query returns the current vertical position of the specified trace.

COMMAND SYNTAX

<trace>:Vert_POSition <display_offset>

<trace> := {TA, TB, TC, TD}

<display_offset> := –5900 DIV to +5900 DIV

Note: The suffix DIV is optional. The limits depend on the current magnification factor, the number of grids on the display, and the initial position of the trace.

QUERY SYNTAX

<trace>:Vert_POSition?

Response format

<trace>:Vert_POSition <display_offset>

EXAMPLE (GPIB)

The following command shifts Trace A (TA) upwards by +3 divisions, relative to the position at the time of acquisition:

```
CMD$="TA:VPOS 3DIV": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

VERT_MAGNIFY

ACQUISITION

VOLT_DIV, VDIV

Command/Query

DESCRIPTION

The VOLT_DIV command sets the vertical sensitivity in Volts/div. The VAB bit (bit 2) in the STB register (see table on page 5–107) is set if an out–of–range value is entered.

Note: The probe attenuation factor is not taken into account for adjusting vertical sensitivity.

The VOLT_DIV? query returns the vertical sensitivity of the specified channel.

COMMAND SYNTAX

<channel>:Volt_DIV <v_gain>

<channel> := {C1, C2, C3*, C4*}

<v_gain> := 2 mV to 5.0 V

Note: The suffix V is optional.

QUERY SYNTAX

<channel>:Volt_DIV?

Response format

<channel>:Volt_DIV <v_gain>

EXAMPLE (GPIB)

The following command sets the vertical sensitivity of channel 1 to 50 mV/div:

```
CMD$="C1:VDIV 50MV":
CALL IBWRT(SCOPE%,CMD$)
```

* 4–channel oscilloscopes only

STATUS

***WAI**
Command

DESCRIPTION

The *WAI (WAIt to continue) command, required by the IEEE 488.2 standard, has no effect on the oscilloscope as the oscilloscope only starts processing a command when the previous command has been entirely executed.

Note: This command can be executed in both local and remote modes.

Command syntax

*WAI

RELATED COMMANDS

*OPC

ACQUISITION

WAIT

Command

DESCRIPTION

The WAIT command prevents the instrument from analyzing new commands until the oscilloscope has completed the current acquisition process.

COMMAND SYNTAX

WAIT

EXAMPLE

```
send: "TRMD SINGLE"
loop {send: "ARM; WAIT;C1:PAVA? MAX"
      read response
      process response
      }
```

This example finds the maximum amplitudes of several signals acquired one after another. ARM starts a new data acquisition. The WAIT command ensures that the maximum is evaluated for the newly acquired waveform.

"C1:PAVA? MAX" instructs the instrument to evaluate the maximum data value in the Channel 1 waveform.

WAVEFORM TRANSFER

WAVEFORM, WF

Command/Query

DESCRIPTION

A WAVEFORM? query transfers a waveform from the oscilloscope to the controller, whereas a WAVEFORM command transfers a waveform from the controller to the oscilloscope.

The WAVEFORM command stores an external waveform back into the oscilloscope's internal memory. A waveform consists of several distinct entities:

(1) the descriptor (DESC),
(2) the user text (TEXT),
(3) the time (TIME) descriptor,
(4) the data (DAT1) block, and optionally
(5) a second block of data (DAT2).

For further information on the structure of the waveform refer to Section 6.

Note: Only complete waveforms queried with "WAVEFORM? ALL" can be restored into the oscilloscope.

The WAVEFORM? query instructs the oscilloscope to transmit a waveform to the controller. The entities may be queried independently. If the "ALL" parameter is specified, all 4 or 5 entities are transmitted in one block in the order enumerated above.

Note: The format of the waveform data depends on the current settings specified by the last WAVEFORM_SETUP command, the last COMM_ORDER command and the last COMM_FORMAT command.

COMMAND SYNTAX

<memory>:WaveForm ALL,<waveform_data_block>

<memory>:= {M1, M2, M3, M4}
<waveform_data_block> := arbitrary data block

QUERY SYNTAX

<trace>:WaveForm? <block>

<trace>:= {TA, TB, TC,TD, M1, M2, M3, M4, C1, C2, C3*, C4*}
<block>:= {DESC, TEXT, TIME, DAT1, DAT2, ALL}

Note: If no parameter is given ALL will be assumed.

Response format

<trace>:WaveForm <block>,<waveform_data_block>

Note: It may be convenient to disable the response header if the waveform is to be restored. Refer to command COMM_HEADER for further details.

* 4–channel oscilloscopes only

EXAMPLE (GPIB)

1) The following command reads the block DAT1 from Memory 1 and saves it in the file "MEM1.DAT". The path header "M1:" is saved together with the data.

```
FILE$ = "MEM1.DAT"
CMD$ = "M1:WF? DAT1"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRDF(SCOPE%,FILE$)
```

2) In the following example, the entire contents of Channel 1 are saved in the file "CHAN1.DAT". The path header "C1:" is skipped to ensure that the data can later be recalled into the oscilloscope.

```
FILE$ = "CHAN1.DAT":RD$=SPACE$(3)
CMD$ = "CHDR SHORT; C1:WF?"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$) Skip first 3 characters "C1:"
CALL IBRDF(SCOPE%,FILE$) Save data in the file
                                    "CHAN1.DAT"
```

3) The following example illustrates how the waveform data saved in example 2) can be recalled into Memory C.

```
FILE$ = "CHAN1.DAT"
CMD$ = "M1:"
CALL IBWRT(SCOPE%,CMD$)
CALL IBWRTF(SCOPE%,FILE$)
```

The "M1:" command ensures that the active waveform is "M1". When the data file is sent to the instrument, it first sees the header "WF" (the characters "C1:" having been skipped when reading the file) and assumes the default destination "M1".

RELATED COMMANDS

INSPECT?, COMM_FORMAT, COMM–ORDER, FUNCTION_STATE, TEMPLATE?, WAVEFORM_SETUP, WAVEFORM_TEXT

WAVEFORM TRANSFER

WAVEFORM_SETUP, WFSU

Command/Query

DESCRIPTION

The WAVEFORM_SETUP command specifies the amount of data in a waveform which will be transmitted to the controller. The command controls the settings of the following parameters:

a. Sparsing (SP). The sparsing parameter defines the interval between data points. For example:

SP =	0	sends all data points
SP =	1	sends all data points
SP =	4	sends every 4th data point

b. Number of points (NP). The number of points parameter indicates how many points should be transmitted. For example:

NP =	0	sends all data points
NP =	1	sends 1 data point
NP =	50	sends a maximum of 50 data points
NP =	1001	sends a maximum of 1001 data points

c. First point (FP). The first point parameter specifies the address of the first data point to be sent. For waveforms acquired in sequence mode, this refers to the relative address in the given segment. For example:

FP =	0	corresponds to the first data point
FP =	1	corresponds to the second data point
FP =	5000	corresponds to data point 5001

d. Segment number (SN). The segment number parameter indicates which segment should be sent if the waveform was acquired in sequence mode. This parameter is ignored for non–segmented waveforms. For example:

SN =	0	all segments
SN =	1	first segment
SN =	23	segment 23

The WAVEFORM_SETUP? query returns the transfer parameters currently in use.

Notation

FP	first point	NP	number of points
SN	segment number	SP	sparsing

COMMAND SYNTAX

WaveForm_SetUp SP,<sparsing>,NP,<number>,FP,<point>, SN,

	L models	M models	Others
<sparsing> :=	0 to 500000 0 = no sparsing	0 to 25000	0 to 5000
<number> :=	0 to 1000000 0 = all points	0 to 50000	0 to 10000
<point> :=	0 to 1000000	0 to 50000	0 to 10000
:=	0 to 2000 0 = all segments	0 to 200	0 to 50

Note 1: After power–on, all values are set to 0 (i.e. entire waveforms will be transmitted without sparsing).0

Note 2: Parameters are grouped in pairs. The first one names the variable to be modified and the second one gives the new value to be assigned. Pairs may be given in any order and may be restricted to those variables to be changed.

QUERY SYNTAX

WaveForm_SetUp?

Response format

WaveForm_SetUp SP,<sparsing>, NP,<number>, FP,<point>,SN,

EXAMPLE (GPIB)

The following command specifies that every 3rd data point (SP=3) starting at address 200 should be transferred:

```
CMD$="WFSU SP,3,FP,200"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INSPECT?, WAVEFORM, TEMPLATE

WAVEFORM TRANSFER

WAVEFORM_TEXT, WFTX

Command/Query

DESCRIPTION

The WAVEFORM_TEXT command is used to document the conditions under which a waveform has been acquired. The text buffer is limited to 160 characters.

The WAVEFORM_TEXT? query returns the text section of the specified trace.

COMMAND SYNTAX

<trace>:WaveForm_TeXt '<text>'

<trace> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3*, C4*}
<text> := An ASCII message (max. 160 characters long)

QUERY SYNTAX

<trace>:WaveForm_TeXt?

Response format

<trace>:WaveForm_TeXt "<text>"

EXAMPLE (GPIB)

The following example shows how to document Trace A (TA):

```
MSG$ = "'Averaged pressure signal.
       Experiment carried out Jan.15, 92'"
CMD$ = "TA:WFTX "+ MSG$
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INSPECT?, WAVEFORM, TEMPLATE?

* 4–channel oscilloscopes only

DISPLAY

XY_ASSIGN?, XYAS?

Query

DESCRIPTION

The XY_ASSIGN? query returns the traces currently assigned to the XY display. If there is no trace assigned to the X–axis and/or the Y–axis the value UNDEF will be returned instead of the trace name.

QUERY SYNTAX

XY_ASsign?

Response Format

XY_ASsign <X_source>,<Y_source>

<X_source> := {UNDEF, TA, TB, TC, TD, C1, C2, C3*, C4*}

<Y_source> := {UNDEF, TA, TB, TC, TD, C1, C2, C3*, C4*}

EXAMPLE (GPIB)

The following query finds the traces assigned to the X–axis and the Y–axis respectively:

CMDS$="XYAS?": CALL IBWRT(SCOPE%,CMDS$)

Example of response message:
XYAS C1,C2

RELATED COMMANDS

TRACE

* 4–channel oscilloscopes only

CURSOR

XY_CURSOR_ORIGIN, XYCO

Command/Query

DESCRIPTION

The XY_CURSOR_ORIGIN command sets the position of the origin for absolute cursor measurements on the XY display.

Absolute cursor values may be measured either with respect to the point (0,0) volts (OFF) or with respect to the center of the XY grid (ON).

The XY_CURSOR_ORIGIN query returns the current assignment of the origin for absolute cursor measurements.

COMMAND SYNTAX

XY_Cursor_Origin <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

XY_Cursor_Origin?

Response Format

XY_Cursor_Origin <mode>

EXAMPLE (GPIB)

The following command sets the origin for absolute cursor measurements to the center of the XY grid.

```
CMDS$="XYCO ON": CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

XY_CURSOR_VALUE

CURSOR

XY_CURSOR_SET, XYCS

Command/Query

DESCRIPTION

The XY_CURSOR_SET command allows the user to position any one of the six independent XY voltage cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed or if the XY display mode is OFF.

The XY_CURSOR_SET? query indicates the current position of the cursor(s).

The CURSOR_SET command is used to position the time cursors.

Notation	
XABS	Vertical absolute on X axis
XREF	Vertical reference on X axis
XDIF	Vertical difference on X axis
YABS	Vertical absolute on Y axis
YREF	Vertical reference on Y axis
YDIF	Vertical difference on Y axis

COMMAND SYNTAX

XY_Cursor_Set <cursor>,<position>

[<cursor>,<position> ....<cursor>,<position>]

<cursor> := {XABS, XREF,XDIF, YABS, YREF, YDIF}

<position> := – 4 to 4 DIV

Note 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first one names the cursor to be modified and the second one indicates its new value. Pairs may be given in any order and may be restricted to those items to be changed.

QUERY SYNTAX

XY_Cursor_Set? [<cursor,...<cursor>]

<cursor> := {XABS, XREF, XDIF, YABS, YREF, YDIF, ALL}

Note: If <cursor> is not specified, ALL will be assumed.

Response Format

XY_Cursor_Set <cursor>,<position>[<cursor>,<position>...

...,<cursor>,<position>]

EXAMPLE (GPIB)

The following command positions the XREF and YDIF at +3 DIV and –2 DIV respectively.

```
CMDS$="XYCS XREF,3DIV,YDIF,–2DIV"
CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

XY_CURSOR_VALUE, CURSOR_SET

CURSOR

XY_CURSOR_VALUE?, XYCV?

Query

DESCRIPTION

The XY_CURSOR_VALUE? query returns the current values of the X versus Y cursors. The X versus Y trace does not need to be displayed to obtain these parameters, but valid sources must be assigned to the X and Y axes.

Notation	
<cursor_type> := [HABS, HREL, VABS, VREL]	
<cursor type>_X	X
<cursor type>_Y	Y
<cursor type>_RATIO	ΔY/ΔX
<cursor type>_PROD	ΔY*ΔX
<cursor type>_ANGLE	arc tan(ΔY/ΔX)
<cursor type>_RADIUS	sqrt(ΔX*ΔX + ΔY*ΔY)

QUERY SYNTAX

XY_Cursor_Value? [<parameter>,...<parameter>]

<parameter> := {HABS_X, HABS_Y, HABS_RATIO, HABS_PROD, HABS_ANGLE, HABS_RADIUS, HREL_X, HREL_Y, HREL_RATIO, HREL_PROD, HREL_ANGLE, HREL_RADIUS, VABS_X, VABS_Y, VABS_RATIO, VABS_PROD, VABS_ANGLE, VABS_RADIUS, VREL_X, VREL_Y, VREL_RATIO, VREL_PROD, VREL_ANGLE, VREL_RADIUS, ALL}

Note: If <parameter> is not specified or equals ALL, all the measured cursor values are returned. If the value of a cursor could not be determined in the current environment, the value UNDEF will be returned. If no trace has been assigned to either the X axis or the Y axis, an environment error will be generated.

Response Format

XY_Cursor_Value <parameter>,<value>
[,...<parameter>,<value>]

<value> := decimal value or UNDEF

EXAMPLE (GPIB)

The following query reads the ratio of the absolute horizontal cursor, the angle of the relative horizontal cursor and the product of the absolute vertical cursors.

```
CMDS$="XYCV? HABS_RATIO,HREL_ANGLE,VABS_PROD
CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

CURSOR_VALUE, XY_CURSOR_ORIGIN

DISPLAY

XY_DISPLAY, XYDS

Command/Query

DESCRIPTION

The XY_DISPLAY command enables or disables the XY display mode.

The XY_DISPLAY? query returns the current mode of the XY display.

COMMAND SYNTAX

XY_DiSplay <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

XY_DiSplay?

Response Format

XY_DiSplay <mode>

EXAMPLE (GPIB)

The following command turns the XY display ON.

CMDS$="XYDS ON": CALL IBWRT(SCOPE%,CMDS$)

REMOTE CONTROL INDEX OF GPIB COMMANDS

Full Command	Abbrev.	Page
ALL_STATUS?	ALST?	5–5
ARM_ACQUISITION	ARM	5–6
ATTENUATION	ATTN	5–7
AUTO_CALIBRATE	ACAL	5–8
AUTO_SETUP	ASET	5–9
BANDWIDTH_LIMIT	BWL	5–10
BUZZER	BUZZ	5–11
*CAL?	*CAL?	5–12
CAL_OUTPUT	COUT	5–13
CALL_HOST	CHST	5–14
CLEAR_SWEEPS	CLSW	5–15
*CLS	*CLS	5–16
CMR?	CMR?	5–17
COMM_FORMAT	CFMT	5–18
COMM_HEADER	CHDR	5–20
COMM_HELP	CHL	5–21
COMM_ORDER	CORD	5–22
COMM_RS232	CORS	5–23
COUPLING	CPL	5–26
CURSOR_MEASURE	CRMS	5–27
CURSOR_SET	CRST	5–29
CURSOR_VALUE?	CRVA?	5–31
DATE	DATE	5–32
DDR?	DDR?	5–33
DEFINE	DEF	5–34
DELETE_FILE	DELF	5–41
DIRECTORY_LIST?	DIR?	5–42
DISPLAY	DISP	5–43
DUAL_ZOOM	DZOM	5–44
*ESE	*ESE	5–45
*ESR?	*ESR?	5–46
EXR?	EXR?	5–48
FORMAT_CARD	FCRD	5–50
FORMAT_FLOPPY	FFLP	5–52
FUNCTION_RESET	FRST	5–54
GRID	GRID	5–55
HARDCOPY_SETUP	HCSU	5–56
HARDCOPY_TRANSMIT	HCTR	5–58
HOR_MAGNIFY	HMAG	5–59
HOR_POSITION	HPOS	5–60
*IDN?	*IDN?	5–62
INE	INE	5–63
INR?	INR?	5–64
INSPECT?	INSP?	5–65
INTENSITY	INTS	5–67
INTERLEAVED	ILVD	5–68
*IST?	*IST?	5–69
KEY	KEY	5–70
MEMORY_SIZE	MSIZ	5–72
MESSAGE	MSG	5–73
MULTI_ZOOM	MZOM	5–74
OFFSET	OFST	5–75
*OPC	*OPC	5–76
*OPT?	*OPT?	5–77
PANEL_SETUP	PNSU	5–78
PARAMETER_CLR	PACL	5–79
PARAMETER_DELETE	PADL	5–80
PARAMETER_STATISTICS?	PAST?	5–81
PARAMETER_VALUE?	PAVA?	5–82
PASS_FAIL_CONDITION	PFCO	5–84
PASS_FAIL_COUNTER	PFCT	5–86
PASS_FAIL_DO	PFDO	5–87
PASS_FAIL_MASK	PFMS	5–89
PERSIST	PERS	5–90
PER_CURSOR_SET	PECS	5–91
PER_CURSOR_VALUE	PECV	5–93
*PRE	*PRE	5–94
*RCL	*RCL	5–95
RECALL	REC	5–96
RECALL_PANEL	RCPN	5–97
*RST	*RST	5–98
SAMPLE_CLOCK	SCLK	5–99
*SAV	*SAV	5–100
SCREEN_DUMP	SCD	5–101
SEGMENTS	SEGS	5–102
SELECT	SEL	5–103
SEQUENCE	SEQ	5–104
*SRE	*SRE	5–105
*STB?	*STB?	5–106
STOP	STOP	5–108
STORE	STO	5–109
STORE_PANEL	STPN	5–110
STORE_SETUP	STST	5–111
STORE_TEMPLATE	STTM	5–112
TEMPLATE?	TMPL?	5–113
TIME_DIV	TDIV	5–114
TRACE	TRA	5–116
*TRG	*TRG	5–117
TRIG_COUPLING	TRCP	5–118
TRIG_DELAY	TRDL	5–119
TRIG_LEVEL	TRLV	5–120
TRIG_MODE	TRMD	5–121
TRIG_SELECT	TRSE	5–122
TRIG_SLOPE	TRSL	5–125
TRIG_WINDOW	TRWI	5–126
*TST?	*TST?	5–127
URR?	URR?	5–128
VERT_MAGNIFY	VMAG	5–129
VERT_POSITION	VPOS	5–130
VOLT_DIV	VDIV	5–131
*WAI	WAI	5–132
WAIT	WAIT	5–133
WAVEFORM	WF	5–134
WAVEFORM_SETUP	WFSU	5–136
WAVEFORM_TEXT	WFTX	5–138
XY_ASSIGN?	XYAS?	5–139
XY_CURSOR_ORIGIN	XYCO	5–140
XY_CURSOR_SET	XYCS	5–141
XY_CURSOR_VALUE?	XYCV?	5–143
XY_DISPLAY	XYDS	5–144

INTRODUCTION

This section discusses how to read and write waveforms and understand their contents. Waveforms can be divided into two basic entities, the basic data array (i.e. the raw data values from the ADC's in the acquisition) and the accompanying descriptive information, such as vertical scale, horizontal scale, time of day, which are necessary for a full understanding of the data.

The information in a waveform can be accessed using the INSPECT? query which interprets it in an easily understood ASCII text form. It can also be more rapidly transferred using the WAVEFORM? query or written back into the instrument with the WAVEFORM command. The oscilloscope contains a data structure called the template which is a detailed description of how the waveform's information is organized.

LOGICAL DATA BLOCKS OF A WAVEFORM

The template gives a detailed description of the form and contents of the logical data blocks of a waveform. It is provided as a reference to be used by you and your programs. A sample template is given in Appendix B although you are encouraged to use the TEMPLATE? query to examine the actual template that your instrument is using. The template may change as the instrument's firmware is enhanced. The template will help provide backward compatibility for the interpretation of waveforms.

Usually, a waveform will contain just a Waveform descriptor block (1.) and a Data array block (5.). In more complicated cases one or more of the other blocks will be present. The data blocks are:

1. **Waveform descriptor block (WAVEDESC)**. This block includes all the information necessary to reconstitute the display of the waveform from the data. This includes:
   - hardware settings at the time of acquisition
   - the exact time of the event
   - the kinds of processing that have been performed
   - the name and serial number of the instrument
   - the encoding format used for the data blocks
   - miscellaneous constants
2. **An optional user–provided text (USERTEXT)**. The WFTX command can be used to put a title or description of a waveform into this block. The WFTX? query command gives an alternative way to read it. This text block can hold up to 160 characters. They can be displayed in the TEXT + TIMES status menu as 4 lines of 40 characters.
3. **A block of sequence acquisition times (TRIGTIME)**. This block is needed for sequence acquisitions to record the exact

timing information for each segment. It contains the time of each trigger relative to the trigger of the first segment, as well as the time of the first data point of each segment relative to its trigger.

4. **A block of random interleaved sampling times (RISTIME).** This block is needed for RIS acquisitions to record the exact timing information for each segment.

5. **A data array block (SIMPLE or DATA_ARRAY_1).** This is the basic integer data of the waveform. It can be raw or corrected ADC data or the integer result of waveform processing.

6. **A second data array block (DATA_ARRAY_2).** This second data array is needed to hold the results of processing functions such as the Extrema (WP01 option) or Complex FFT (WP02 option). In such cases, the data arrays contain:

	Extrema	FFT
DATA_ARRAY_1	Roof trace	Real part
DATA_ARRAY_2	Floor trace	Imaginary part

Note: The TEMPLATE also describes an array named DUAL. This is simply a way to allow the INSPECT? command to examine the two data arrays together.

INSPECT? COMMAND

This is the simplest way to examine the contents of a waveform. It can be used on both the data and descriptive parts. The simplest form of the command is:

INSPECT? "name"

where the template gives the name of a descriptor item or data block. The answer is returned as a single string, but may span many lines. Here is some typical dialogue:

```
question    C1:INSPECT? "VERTICAL_OFFSET"
response    C1:INSP "VERTICAL_OFFSET   : 1.5625e-03        "
question    C1:INSPECT? "TRIGGER_TIME"
response    C1:INSP "
            TRIGGER_TIME   : Date = FEB 17, 1992, Time =  4: 4:29.5580
"
```

The INSPECT? command can also be used to get a readable translation of the full waveform descriptor block with the command:

INSPECT? "WAVEDESC"

The template dump from your instrument (or from Appendix B) will give details on the interpretation of each of the parameters.

The INSPECT? command is also used to examine the measured data values of a waveform. For an acquisition with 52 points we get:

```
INSPECT? "SIMPLE"
C1:INSP "
0.0005225       0.0006475       –0.00029        –0.000915       2.25001E–05     0.000835
0.0001475       –0.0013525      –0.00204        –4E–05          0.0011475       0.0011475
–0.000915       –0.00179        –0.0002275      0.0011475       0.001085        –0.00079
–0.00179        –0.0002275      0.00071         0.00096         –0.0003525      –0.00104
0.0002725       0.0007725       0.00071         –0.0003525      –0.00129        –0.0002275
0.0005225       0.00046         –0.00104        –0.00154        0.0005225       0.0012725
0.001335        –0.0009775      –0.001915       –0.000165       0.0012725       0.00096
–0.000665       –0.001665       –0.0001025      0.0010225       0.00096         –0.0003525
–0.000915       8.50001E–05     0.000835        0.0005225
"
```

These numbers are the fully converted measurements in volts. Of course, when the data block contains thousands of items the string will contain many lines.

Depending on the application, you may prefer to have the data in its raw form as either a BYTE (8 bits) or a WORD (16 bits) for each data value. In this case you must use the relations given below in association with the WAVEFORM? command to interpret the measurement. The command might then say:

```
INSPECT? "SIMPLE",BYTE
```

The examination of data values for waveforms with two data arrays can be done as follows:

```
INSPECT? "DUAL"         to get pairs of data values on a single line

INSPECT? "DATA_ARRAY_1"
                        to get the values of the first data array

INSPECT? "DATA_ARRAY_2"
                        to get the values of the second data array
```

It is also possible to examine just a part of the waveform or a sparsed form of the waveform. This is controlled with the WAVEFORM_SETUP command mentioned later in this section.

The INSPECT? command has only a query form. It cannot be used to send a waveform back into the oscilloscope. It is also a very verbose way in which to send the information and is not very fast. Users who need speed or the ability to send the waveform back to the instrument should use the WAVEFORM commands.

BASIC users might find it convenient to combine the capabilities of the inspect facility with the waveform query command in order to construct files containing a human and BASIC readable version of the waveform descriptor together with the full waveform in a format suitable for retransmission to the instrument. This can be done for a waveform in a memory location by sending the command:

```
MC:INSPECT? "WAVEDESC";WAVEFORM?
```

and putting the response directly into a disk file.

WAVEFORM? COMMAND

The WAVEFORM commands are an efficient way to transfer waveform data using the block formats defined in the IEEE–488.2 standard. You have the possibility of reading all of the logical blocks of the waveform with a single query:

```
C1:WAVEFORM?
```

This is the preferred form for most applications since it is complete and the response can be downloaded back into the instrument using the WAVEFORM command. You can also choose to read any single block with a query like:

```
C1:WAVEFORM? DAT1
```

This can save time and space when you need to read many waveforms all with the same acquisition conditions or if you are only interested in lots of raw integer data. Consult the description of the WAVEFORM command in Section 5 for the names of the various blocks.

Please be aware that a waveform query response can easily be a block containing over 200,000 bytes if it is in binary format and twice as much if the HEX option is used.

Interpreting the waveform descriptor

The binary response to a query command of the form:

```
C1:WAVEFORM? or C1:WAVEFORM? ALL
```

can be put into a disk file and then dumped to show the following hexadecimal and ASCII form: (This was done over GPIB with default settings)

Byte offset	Binary contents in hexadecimal	ASCII translation (. is for uninteresting)
0	57 46 20 41 4c 4c 2c 23 39 30 30 30 30 30 30 34	WF ALL,#90000004
16	35 30 57 41 56 45 44 45 53 43 00 00 00 00 00 00	50WAVEDESC.....
32	00 00 4c 45 43 52 4f 59 5f 32 5f 31 00 00 00 00	..LECROY_2_1....
48	00 00 00 01 00 00 00 00 01 5a 00 00 00 00 00 00	.....
64	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
80	00 68 00 00 00 00 00 00 00 00 00 00 00 00 4c 45	.....LE
96	43 52 4f 59 39 33 31 34 4c 00 00 00 00 00 37 84	CROY9314L.....
112	09 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
128	00 00 00 00 00 34 00 00 00 34 00 00 00 32 00 00	
144	00 00 00 00 00 33 00 00 00 00 00 00 00 01 00 00	
160	00 00 00 00 00 01 00 00 00 01 00 00 00 00 34 83	
176	12 6f 3a 0d 8e c9 46 fe 00 00 c7 00 00 00 00 08	
192	00 01 32 2b cc 77 be 6b a4 bb 51 a0 69 bb be 6a	
208	d7 f2 a0 00 00 00 56 00 00 00 00 00 00 00 00 00	
224	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
256	00 00 00 00 00 00 53 00 00 00 00 00 00 00 00 00	
272	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
288	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
304	00 00 00 00 00 00 00 00 00 00 40 3b 00 00 00 00	
320	00 00 17 0a 35 02 07 c8 00 00 00 00 00 00 00 00	
336	00 00 00 00 00 01 00 0e 00 04 3f 80 00 00 00 0a	
352	00 00 3f 80 00 00 3a 0d 8e c9 00 00 11 00 00 00	
368	04 00 fa 00 09 00 16 00 0b 00 f3 00 e8 00 08 00	
384	1b 00 1b 00 fa 00 ec 00 0b 00 1b 00 1a 00 fc 00	
400	ec 00 05 00 14 00 18 00 03 00 f8 00 0d 00 1b 00	
416	14 00 03 00 f4 00 05 00 11 00 10 00 f8 00 f0 00	
432	11 00 1a 00 1e 00 f9 00 ea 00 06 00 1d 00 18 00	
448	fe 00 ee 00 07 00 19 00 18 00 03 00 fa 00 0a 00	
464	16 00 11 00 0a	

It can be seen that the first 7 bytes translate into ASCII and look like the simple beginning of a query response. This is followed by the string "#9000000450". This is the beginning of a binary block where 9 ASCII integers are used to give the length of the block (450 bytes). The waveform itself starts immediately after this at byte number 18 (the first byte is byte 0).

Deciphering the waveform descriptor can be done with the aid of the template (see Appendix B). It states that the first object is a DESCRIPTOR_NAME which is a string of 16 characters with the value WAVEDESC and this is what we see. At byte 16 relative to the beginning of the descriptor (or byte 34 above) we find the next string, the TEMPLATE_NAME with the value LECROY_2_1. Several other parameters follow. We can easily recognize the INSTRUMENT_NAME at 76 bytes from the descriptor start (or byte 94 above).

In a similar way we learn that a 4–byte–long integer giving the length of the descriptor starts at byte 36 (or byte 54 above):

WAVE_DESCRIPTOR = 15a (hex) = 346

At byte 60 (or byte 78 above) we find another 4–byte integer giving the length of the data array:

WAVE_ARRAY_1 = 68 (hex) = 104

and at byte 116 (or byte 134 above) the number of data points:

WAVE_ARRAY_COUNT = 34 (hex) = 52

Now we know that the data will start at byte 346 from the beginning of the descriptor (or byte 364 above) and that each of the 52 data points will be represented by two bytes. The waveform has a total length of 346 + 104 which is the same as the ASCII string told us at the beginning of the block. The final 0a at byte 468 is the NL character associated with the GPIB message terminator <NL><EOI>.

The data can be easily seen starting at byte 364 above. Since the example was taken using an oscilloscope with an 8–bit ADC we see those 8 bits followed by a 0 byte for each data point. It should be noted that for many other kinds of waveform, this second byte will not be zero and contains interesting information. The data is coded in signed form (two's complement) with values ranging from –32768 = 8000 (hex) to 32767 = 7fff (hex). If we had chosen to use the BYTE option for the data format the values would have been signed integers in the range –128 = 80 (hex) to 127 = 7f (hex).

Interpreting the waveform vertical data

Now that we know how to decipher the data it would be useful to convert it to the appropriate measured values. The vertical reading for each data point depends on the vertical gain and the vertical offset given in the descriptor. For acquisition waveforms this corresponds to the volts/div and voltage offset selected after conversion for the data representation being used. The template tells us that the vertical gain and offset can be found at bytes 156 and 160 respectively of the descriptor and that they are stored as floating point numbers in the IEEE 32 bit format. An ASCII string giving the vertical unit is to be found in VERTUNIT, byte 196. The vertical value is given by the relationship:

value = VERTICAL_GAIN * data – VERTICAL_OFFSET

A data value of 0 is normally displayed as a point in the middle of the grid.

In the case of the data shown above we find:

VERTICAL_GAIN = 2.44141e–07 from the floating point number 3483 126f at byte 174

VERTICAL_OFFSET = 0.00054 from the floating point number 3a0d 8ec9 at byte 178

VERTICAL_UNIT = V = volts from the string 5600 ... at byte 214

and therefore:

since data[0] = 4352 from the hexadecimal word 1100 at byte 364

value[0] = 0.0005225 V as stated in the inspect command above

and

since data[1] = 4864 from the hexadecimal word 1300 at byte 366

value[1] = 0.0006475 V as stated in the inspect command above.

If your computer or available software is incapable of understanding the IEEE floating point values you can find a description of this format in the template (see Appendix B).

The data values in a waveform may not all correspond to measured points. The parameters, FIRST_VALID_PNT and LAST_VALID_PNT give the necessary information. The descriptor also records the SPARSING_FACTOR, the FIRST_POINT, and the SEGMENT_INDEX to aid interpretation if the options of the WAVEFORM_SETUP command have been used.

For sequence acquisitions the data values for each segment are given in their normal order and the segments are read out one after the other. The important descriptor parameters are the WAVE_ARRAY_COUNT and the SUBARRAY_COUNT, giving the total number of points and the number of segments.

For waveforms such as the extrema and the complex FFT there will be two arrays one after the other for the two arrays of the result.

Calculating the horizontal position of a data point

Each vertical data value has a corresponding horizontal position, usually measured in time or in frequency units. The calculation of this position depends on the type of waveform being examined. We will treat separately the single sweep, the sequence, and the interleaved (RIS) waveform. Each data value has a position, i, in the original waveform with i = 0 corresponding to the first data point

acquired. The descriptor parameter HORUNIT gives a string with the name of the horizontal unit.

• Single–sweep waveforms

x[i] = HORIZ_INTERVAL * i + HORIZ_OFFSET

For acquisition waveforms this time is from the trigger to the data point in question. It will be different from acquisition to acquisition since the HORIZ_OFFSET is measured for each trigger.

In the case of the data shown above this means:

HORIZ_INTERVAL = 1e–08 from the floating point number 322b cc77 at byte 194

HORIZ_OFFSET = –5.149e–08 from the double precision floating point number be6b a4bb 51a0 69bb at byte 198

HORUNIT = S = seconds from the string 5300 ... at byte 262

which gives

x[0] = –5.149e–08 S
x[1] = –4.149e–08 S

• Sequence waveforms

Since sequence waveforms are really many independent acquisitions, each segment will have its own horizontal offset. These can be found in the TRIGTIME array. For the n'th segment

x[i,n] = HORIZ_INTERVAL * i + TRIGGER_OFFSET[n]

The TRIGTIME array can contain up to 200 segments of timing information with two 8 byte double precision floating point numbers for each segment.

• Interleaved (RIS) waveforms

These waveforms are composed of many acquisitions interleaved together. The descriptor parameter, SWEEPS_PER_ACQ gives the number of acquisitions. The i'th point will belong to the m'th segment where

m = i modulo (SWEEPS_PER_ACQ)

will have a value between 0 and SWEEPS_PER_ACQ – 1.

Then with

j = i – m

x[i] = x[j,m] = HORIZ_INTERVAL * j + RIS_OFFSET[m]

where the RIS_OFFSET's can be found in the RISTIME array. There can be up to 100 8 byte double precision floating point numbers in this block. The instrument tries to get segments with times such that

RIS_OFFSET[i] ≈ PIXEL_OFFSET + (i – 0.5) * HORIZ_INTERVAL

Thus, taking as an example a RIS with SWEEPS_PER_ACQ = 10 HORIZ_INTERVAL = 1 ns and PIXEL_OFFSET = 0.0, we might find for a particular event that:

RIS_OFFSET[0] =–0.5 ns	RIS_OFFSET[1] = 0.4 ns
RIS_OFFSET[2] = 1.6 ns	RIS_OFFSET[3] = 2.6 ns
RIS_OFFSET[4] = 3.4 ns	RIS_OFFSET[5] = 4.5 ns
RIS_OFFSET[6] = 5.6 ns	RIS_OFFSET[7] = 6.4 ns
RIS_OFFSET[8] = 7.6 ns	RIS_OFFSET[9] = 8.5 ns

and therefore:

x[0] =	RIS_OFFSET[0]	=	–0.5 ns
x[1] =	RIS_OFFSET[1]	=	0.4 ns
...			
x[9] =	RIS_OFFSET[9]	=	8.5 ns
x[10] =	1 ns * 10 + (–0.5)	=	9.5 ns
x[11] =	1 ns * 10 + 0.4	=	10.4 ns
...			
x[19] =	1 ns * 10 + 8.5	=	18.5 ns
x[20] =	1 ns * 20 + (–0.5)	=	19.5 ns
...			

WAVEFORM COMMAND

Waveforms that have been read in their entirety with the WAVEFORM? command can be sent back into the instrument. Since the descriptor contains all of the necessary information, you do not have to be careful about any of the communication format parameters. The instrument can learn all that it needs to know from the waveform.

If you want to synthesize waveforms for display or comparison purposes, you are encouraged to read out a waveform of the appropriate size and then replace the data with the desired values. This will assure that the descriptor is coherent.

Note: You are only allowed to send back waveforms to memory traces (M1, M2, M3, M4). This means that you may have to remove or change the prefix (C1 or CHANNEL_1) in the response to the

WF? query. The examples for the WF command in Section 5 show how this can be done.

MORE CONTROL OF WAVEFORM QUERIES

There are many different ways for you to use the WAVEFORM? command which may simplify or speed up your work. Among them are:

• Partial readout of waveform

The WAVEFORM_SETUP command allows you to specify a short part of a waveform for readout. It also lets you select a sparsing factor to read only every n'th data point.

• Byte swapping

The COMM_ORDER command allows you to swap the two bytes of a 16–bit word. In fact, byte swapping is done for all numbers represented by more than one byte. This is the case for the descriptor, the time blocks, and WORD arrays, thereby simplifying data interpretation for some computer systems (e.g. INTEL based, or DEC)

• Data length, block format, and encoding

The COMM_FORMAT command gives you control over these parameters. If you do not need the extra precision of the lower order byte of the standard data value, the BYTE option lets you save a factor of two on the amount of data to be transmitted or stored. If your computer is not able to read binary data, the HEX option allows a response form where the value of each byte is given by a pair of hexadecimal digits.

• Data only transfers

The COMM_HEADER OFF mode will allow you to get a response to WF? DAT1 with the data only (the C1:WF DAT1 will disappear).

If you have also specified COMM_FORMAT OFF,BYTE,BIN, you will just get a response of data bytes (the #90000nnnnn will disappear).

• Formatting for RS–232 users

The COMM_RS232 command can help you by splitting the very long WF? response into individual lines

HIGH–SPEED WAVEFORM TRANSFER

In order to achieve the maximum continuous data transfer rates from the oscilloscope to your instrument you will have to optimize many factors. The single most important point is to limit the work done in your computer. This means avoiding having to write the data

to disk, minimizing the per data point computations, minimizing the number of calls to the IO system, etc. You can let the instrument help by reducing the number of points to be transferred and the number of data bytes per point. The pulse parameter capability and the processing functions can save you lots of computing and lots of data transfer time if employed creatively. Two other very important principles are:

- Try to overlap waveform acquisition with waveform transfer. The oscilloscope is capable of transferring an already acquired or processed waveform after a new acquisition has been started. This can also considerably increase the total time that the oscilloscope will be able to acquire events if it has to wait for triggers (livetime).

- Minimize the number of waveform transfers by using the sequence mode to accumulate many triggers for each transfer. This is preferable to using the WAVEFORM_SETUP command to reduce the number of data points to be transferred. It also reduces the oscilloscope transfer overhead significantly.

Here is an example of the type of commands to be given:

ARM	to acquire the first event or sequence
WAIT;ARM;C1:WF?	to wait for the event, start the next acquisition and then transfer the data.

This second line can be repeated by your program as soon as it has finished reading the waveform.

An extensive set of status registers allows the user to quickly determine the oscilloscope's internal processing status at any time. The status registers as well as the status reporting system have been designed to comply with IEEE 488.2 recommendations.

Related functions are grouped together in common status registers. Some, such as the Status Byte Register (STB) or the Standard Event Status Register (ESR), are required by the IEEE 488.2 standard. However, other registers are device specific. They include the Command Error Register (CMR) or the Execution Error Register (EXR). Commands associated with IEEE 488.2 mandatory status registers are preceded by an asterisk <*>.

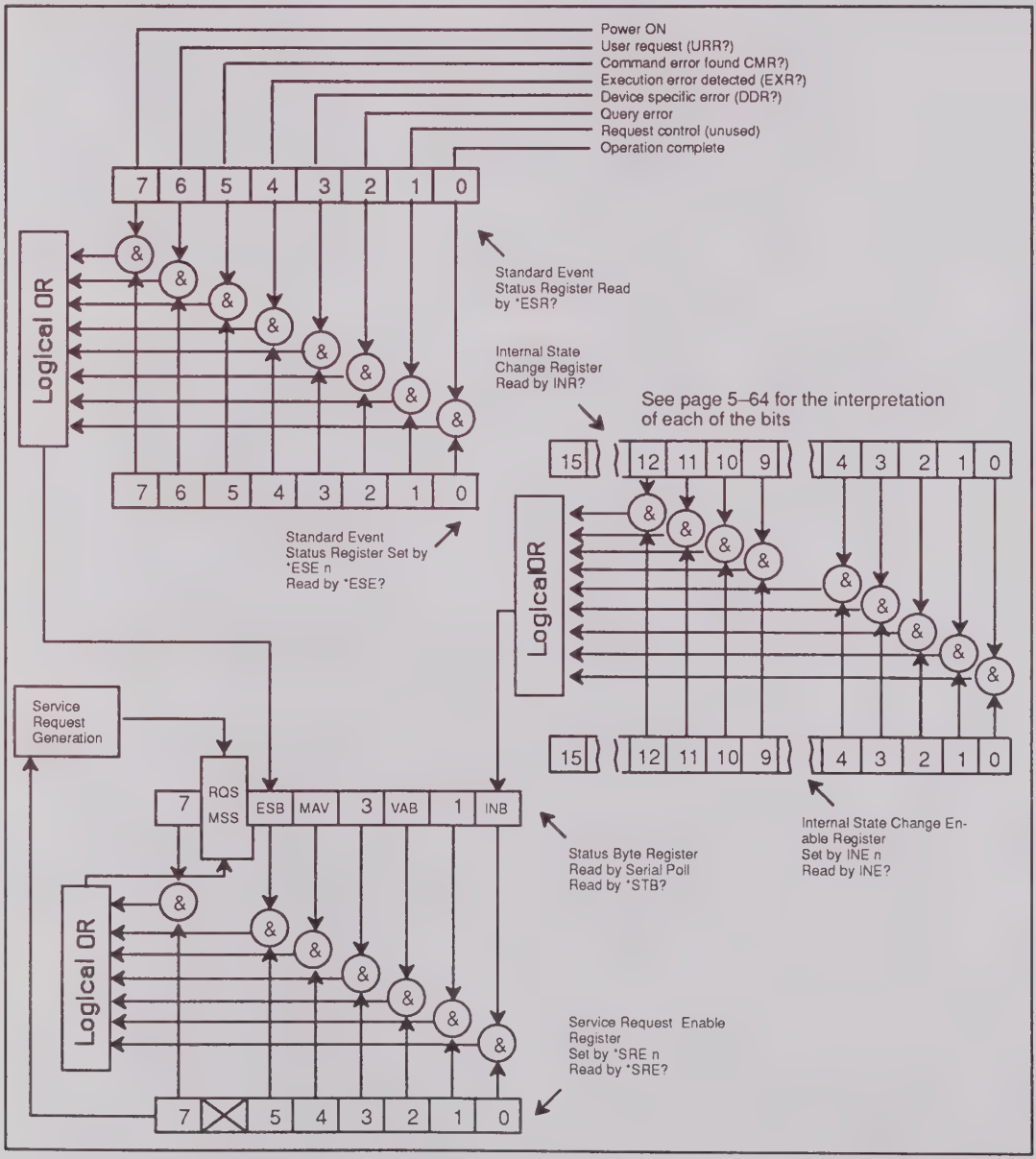
OVERVIEW OF STATUS AND SERVICE REQUEST REPORTING

The Standard Event Status Bit (ESB) and the Internal Status Change Bit (INB) in the Status Byte Register are summary bits of the Standard Event Status Register (ESR) and the Internal State Change Register (INR). The Message Available Bit (MAV) is set whenever there are data bytes in the output queue. The Value Adapted Bit (VAB) indicates that a parameter value was adapted during a previous command interpretation (e.g. if the command "TDIV 2.5 US" is received, the time base is set to 2 µsec/div and the VAB bit is set).

The Master Summary Status bit (MSS) indicates that the instrument requests service. The MSS bit can only be set if any of the other bits of STB are enabled with the Service Request Enable Register (SRE).

All Enable registers (SRE, ESE and INE) are used to generate a bitwise AND with their associated status registers. The logical OR of this operation is reported to the STB register. At power–on, all Enable registers are zero, inhibiting any reporting to the STB.

The Standard Event Status Register (ESR) mostly summarizes errors, whereas the Internal State Change Register (INR) reports internal changes in the instrument. Additional details of the errors reported by ESR can be obtained with the queries "CMR?", "DDR?", "EXR?" and "URR?".


Power ON
User request (URR?)
Command error found CMR?)
Execution error detected (EXR?)
Device specific error (DDR?)
Query error
Request control (unused)
Operation complete
7 6 5 4 3 2 1 0
Logical OR
Standard Event Status Register Read by *ESR?
Internal State Change Register Read by INR?
See page 5–64 for the interpretation of each of the bits
15 12 11 10 9 4 3 2 1 0
Standard Event Status Register Set by *ESE n Read by *ESE?
Logical OR
Service Request Generation
7 RQS MSS ESB MAV 3 VAB 1 INB
Status Byte Register Read by Serial Poll Read by *STB?
Internal State Change Enable Register Set by INE n Read by INE?
Logical OR
Service Request Enable Register Set by *SRE n Read by *SRE?
7 5 4 3 2 1 0

STATUS REGISTER STRUCTURE

The register structure contains one more register (not shown in the above figure). It is the Parallel Poll Enable Register (PRE) which acts exactly like the Service Request Enable Register (SRE), but it sets the "ist" bit (not shown in the above figure), used in the Parallel Poll. The "ist" bit can also be read with the "*IST?" query.

Example of status reporting

If an erroneous remote command, e.g. "TRIG_MAKE SINGLE", is transmitted to the instrument, it rejects the command and sets the Command Error Register (CMR) to the value 1 (unrecognized command/query header). The non–zero value of CMR is reported to bit 5 of the Standard Event Status Register (ESR) which is then set.

Nothing further happens unless the corresponding bit 5 of the Standard Event Status Enable Register (ESE) is set (with the command "*ESE 32"), enabling the fact that bit 5 of ESR is set to be reported to the summary bit ESB of the Status Byte Register (STB).

If setting of the ESB summary bit in STB is enabled, again nothing happens unless further reporting is enabled by setting the corresponding bit in the Service Request Enable Register (with the command "*SRE 32"). In this case, the generation of a non–zero value of CMR ripples through to the Master Summary Status bit (MSS), generating a Service Request (SRQ).

The value of CMR can be read and simultaneously reset to zero at any time with the command "CMR?". The occurrence of a command error can also be detected by analyzing the response to "*ESR?". However, if several types of potential errors must be surveyed, it is usually much more efficient to enable propagation of the errors of interest into the STB with the enable registers ESE and INE.

Summary

A command error (CMR) sets bit 5 of ESR:

- if bit 5 of ESE is set, ESB of STB is also set.
- if bit 5 of SRE is set, MSS/RQS of STB is also set and a Service Request is generated.

STATUS BYTE REGISTER (STB)

The Status Byte Register is the instrument's central reporting structure. The STB is composed of 8 single–bit summary messages (of which 3 are unused) which reflect the current status of the associated data structures implemented in the instrument.

Bit 0 is the summary bit INB of the Internal State Change Register. It is set if any of the bits of the INR are set, provided that they are enabled by the corresponding bit of the INE register.

Bit 2 is the Value Adapted Bit, indicating that a parameter value was adapted during a previous command interpretation.

Bit 4 is the Message Available (MAV) bit, indicating that the interface output queue is not empty.

Bit 5 of STB is the summary bit ESB of the Standard Event Status Register. It is set if any of the bits of the ESR are set, provided that they are enabled by the corresponding bit of the ESE register.

Bit 6 of the Status Byte Register (STB) is alternatively called the Master Summary Status bit (MSS) or the Request for Service bit (RQS) because the STB can be read in two different ways. The command "*STB?" reads and clears the STB in the query mode, in which case bit 6 of the STB is the MSS bit, indicating if the instrument has any reason for requesting service. The other way of reading the STB is the serial poll (see page 3–12 for the GPIB serial poll procedure). In this case, bit 6 of the STB is the RQS bit, indicating that the instrument has actually activated the SRQ line on the GPIB. The serial poll only clears the RQS bit. Therefore, the MSS bit of the STB (and any other bits which caused MSS to be set) will stay set after a serial poll. The controller must reset these bits.

The Status Byte Register may be read via the query "*STB?". The response represents the binary weighted sum of the register bits. The register is cleared by "*STB?", "ALST?", "*CLS" or after the instrument has been powered up.

STANDARD EVENT STATUS REGISTER (ESR)

The ESR is a 16–bit register reflecting the occurrence of events. The register bit assignments have been standardized by IEEE 488.2. Only the lower 8 bits are currently in use.

The Standard Event Status Register may be read via the query "*ESR?". The response is the binary weighted sum of the register bits. The register is cleared with an "*ESR?" or "ALST?" query, a "*CLS" command or after power–on.

Example

The response message "*ESR 160" indicates that a command error occurred and that the ESR is being read the first time after power–on. The value 160 can be broken down into 128 (bit 7) plus 32 (bit 5). See the table on page 5–47 for a description of the conditions corresponding to the bits set.

The "Power ON" bit appears only on the first "*ESR?" query after power–on because the query clears the register. The type of command error can be determined by reading the Command Error Status Register with the query "CMR?". Note that it is not necessary to read (and simultaneously clear) this register in order to be able to set the CMR bit in the ESR on the next command error.

STANDARD EVENT STATUS ENABLE REGISTER (ESE)

The ESE allows one or more events in the Standard Event Status Register to be reported to the ESB summary bit in the STB.

The Standard Event Enable Register is modified with the command "*ESE". It is cleared with the command "*ESE 0", or after power–on. It may be read with the query "*ESE?".

Example

"*ESE 4" sets bit 2 (i.e. binary 4) of the standard event enable register, enabling query errors to be reported.

SERVICE REQUEST ENABLE REGISTER (SRE)

The Service Request Enable Register specifies which summary bit(s) in the Status Byte Register will cause a service request. The Service Request Enable Register consists of 8 bits. Setting a bit in the register allows the summary bit located at the same bit position in the Status Byte Register to generate a service request provided that the associated event becomes true. Bit 6 (MSS) cannot be set and is always reported as zero in response to the query "*SRE?".

The Standard Event Enable Register is modified with the command "*SRE". It is cleared with the command "*SRE 0", or after power–on. It may be read with the query "*SRE?".

PARALLEL POLL ENABLE REGISTER (PRE)

The Parallel Poll Enable Register specifies which summary bit(s) in the Status Byte Register will set the "ist" individual local message. This register is quite similar to the Service Request Enable Register (SRE), but it is used to set the parallel poll "ist" bit rather than MSS.

The value of the "ist" may also be read without a Parallel Poll via the query "*IST?". The response indicates if the "ist" message has been set or not (values are 1 or 0).

The Parallel Poll Enable Register is modified with the command "*PRE". It is cleared with the command "*PRE 0", or after power–on. It may be read with the query "*PRE?". (See page 3–13 for the GPIB parallel poll procedure.)

Example

"*PRE 5" sets bits 2 and 0 (decimal 4 and 1) of the Parallel Poll Enable Register.

INTERNAL STATE CHANGE STATUS REGISTER (INR)

The INR reports the completion of a number of internal operations. The events tracked by this 16–bit–wide register are listed with the command "INR?" in Section 5.

The Internal State Change Status Register may be read via the query "INR?". The response is the binary weighted sum of the register bits. The register is cleared with an "INR?" or "ALST?" query, a "*CLS" command or after power–on.

INTERNAL STATE CHANGE ENABLE REGISTER (INE)

The INE allows one or more events in the Internal State Change Status Register to be reported to the INB summary bit in the STB.

The Internal State Change Enable Register is modified with the command "INE". It is cleared with the command "INE 0", or after power–on. It may be read with the query "INE?".

COMMAND ERROR STATUS REGISTER (CMR)

The Command Error Status register contains the code of the last command error detected by the instrument. Command error codes are listed with the command "CMR?" (see page 5–17).

The Command Error Status Register may be read via the query "CMR?". The response is the error code. The register is cleared with a "CMR?" or "ALST?" query, a "*CLS" command or after power–on.

DEVICE DEPENDENT ERROR STATUS REGISTER (DDR)

The DDR indicates the type of hardware errors affecting the instrument. Individual bits in this register report specific hardware failures. They are listed with the command "DDR?" (see page 5–33).

The Device Dependent Error Status Register may be read via the query "DDR?". The response is the binary weighted sum of the error bits. The register is cleared with a "DDR?" or "ALST?" query, a "*CLS" command or after power–on.

EXECUTION ERROR STATUS REGISTER (EXR)

The Execution Error Status Register contains the code of the last execution error detected by the instrument. Execution error codes are listed with the command "EXR?" (see page 5–49).

The Execution Error Status Register may be read via the query "EXR?". The response is the error code. The register is cleared with a "EXR?" or "ALST?" query, a "*CLS" command or after power–on.

USER REQUEST STATUS REGISTER (URR)

The URR contains the identification code of the last menu button which was pressed. The codes are listed with the command "URR?" (see page 5–128).

The User Request Status Register may be read via the query "URR?". The response is the decimal code associated with the selected menu button. The register is cleared with a "URR?" or "ALST?" query, a "*CLS" command or after power–on.

EXAMPLE 1 USE OF THE INTERACTIVE GPIB PROGRAM 'IBIC"

This example assumes the use of an IBM PC, or a compatible computer, equipped with a National Instruments GPIB interface card. It also assumes that the GPIB-driver is left in the default state so that the device name "dev4" corresponds to the GPIB address 4 which is assumed to be the address of the oscilloscope. All text entered by the user is underlined.

ibic<CR>

program announces itself

: ibfind<CR>

enter board/device name: dev4<CR>

dev4: ibwrt<CR>

enter string: "tdiv?"<CR>

[0100] (cmpl)
count: 5

dev4: ibrd<CR>

enter byte count: 10<CR>

[0100] (cmpl)
count: 10
54 44 49 56 20 35 30 45 TDIV 50E
2D 39 -9

dev4: ibwrt<CR>

enter string: "c1:cpl?"<CR>

[0100] (cmpl)

count: 7

dev4: ibrd<CR>

enter byte count: 20<CR>

[2100] (end cmpl)
count: 11
43 31 3A 43 50 4C 20 44 C1:CPL D
35 30 0A 50z

dev4: q<CR>

for quitting the program

EXAMPLE 2 GPIB PROGRAM FOR IBM PC (HIGH-LEVEL FUNCTION CALLS)

The following BASICA program allows full interactive control of the 9420/24/50 using an IBM PC as GPIB controller. It is again assumed that the controller is equipped with a National Instruments GPIB interface card. All the remote control commands listed in Section 5 can be used by simply entering the text string of the command, i.e. "c1:vdiv 50 mv" (without the quotes). The program automatically displays the information sent back by the oscilloscope in response to queries.

In addition, a few utilities have been provided for convenience. The commands ST and RC enable waveform data to be stored on or retrieved from disk if proper drive and file names are provided. The command LC returns the oscilloscope to local mode. Responses sent back by the oscilloscope are interpreted as character strings and are thus limited to a maximum of 255 characters.

Note 1: It is assumed that the National Instruments GPIB driver GPIB.COM is in its default state. This means that the interface board can be referred to by its symbolic name 'GPIB0' and that devices on the GPIB with addresses 1 to 16 can be called by the symbolic name 'DEV1' to 'DEV16'.

Note 2: Lines 1 - 99 are a copy of the file DECL.BAS supplied by National Instruments. The first 6 lines are required for the initialization of the GPIB handler. DECL.BAS requires access to the file BIB.M during the GPIB initialization. BIB.M is one of the files supplied by National Instruments, and must exist in the directory currently in use.

Note 3: The first 2 lines of DECL.BAS each contain a string "XXXXX" which must be replaced by the number of bytes which determine the maximum workspace for BASICA (computed by subtracting the size of BIB.M from the currently available space in BASICA). For example, if the size of BIB.M is 1200 bytes and when BASICA is loaded it reports "60200 bytes free", you would replace "XXXXX" by the value 59000 or less.

Note 4: The default timeout of 10 seconds is modified to 300 msec during the execution of this program. However, the default value of the GPIB handler is not changed. Whenever a remote command is entered by the user, the program sends it to the instrument with the function call IBWRT. Afterwards, it always executes an IBRD call, independently of whether or not a response is expected. If a response is received it is immediately displayed. If there is no response, the program waits until time-out and then asks for the next command.

```
1-99  <DECL.BAS>
100   CLS
110   PRINT "Control of the 9300 via GPIB and IBM PC"
115   PRINT ""
120   PRINT "Options :EX to exit   LC local mode"
125   PRINT "          ST store dataRC recall data"
130   PRINT ""
140   LINE INPUT "GPIB-address of oscilloscope (1...16)? :",ADDR$
145   DEV$ = "DEV" + ADDR$
150   CALL IBFIND(DEV$,SCOPE%)
155   IF SCOPE% < 0 THEN GOTO 830
160   TMO% = 10  'timeout = 300 msec (rather than default 10 sec)
165   CALL IBTMO(SCOPE%,TMO%)
170   '
200   LOOP% = 1
205   WHILE LOOP%
210       LINE INPUT "Enter command (EX --> Exit) : ",CMD$
220       IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP% = 0 : GOTO 310
230       IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 600 : GOTO 300
240       IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 700 : GOTO 300
250       IF CMD$ = "lc" OR CMD$ = "LC" THEN GOSUB 400 : GOTO 300
260       IF CMD$ = "" THEN GOTO 300
270       CALL IBWRT(SCOPE%,CMD$)
275       IF IBSTA% < 0 THEN GOTO 840
280       GOSUB 500
300   WEND
310   GOSUB 400
320   END
400   '
405   'SUBROUTINE LOCAL_MODE
410   '
420   CALL IBLOC(SCOPE%)
425   PRINT ""
430   RETURN
500   '
505   'SUBROUTINE GET_DATA
510   'If there are no data to read, simply wait until timeout occurs
515   '
520   CALL IBRD(SCOPE%,RD$)
525   I = IBCNT% 'IBCNT% is the number of characters read
530   FOR J = 1 TO I
535       PRINT MID$(RD$,J,1);
540   NEXT J
545   PRINT ""
550   RETURN
600
```

```
605   'SUBROUTINE STORE_DATA
610   '
615   RD1$=SPACE$(3)
620   LINE INPUT "Specify trace (TA...TD,M1...M4,C1...C4): ",TRACE$
625   LINE INPUT "Enter filename : ",FILE$
630   CMD$="WFSU NP,0,SP,0,FP,0,SN,0; CHDR SHORT"
640   CALL IBWRT(SCOPE%,CMD$)
645   CMD$=TRACE$+":WF?"
650   CALL IBWRT(SCOPE%,CMD$)
660   CALL IBRD(SCOPE%,RD1$)      'Discard first 3 chars of response
665   CALL IBRDF(SCOPE%,FILE$)
670   IF IBSTA% < 0 THEN GOTO 840
675   PRINT ""
680   RETURN
700   '
705   'SUBROUTINE RECALL_DATA
710   '
715   LINE INPUT "Specify target memory (M1...M4): ",MEM$
720   LINE INPUT "Enter filename : ",FILE$
730   CMD$=MEM$+":"
735   CALL IBWRT(SCOPE%,CMD$)
740   CALL IBWRTF(SCOPE%,FILE$)
745   IF IBSTA% < 0 THEN GOTO 840
750   PRINT ""
755   RETURN
800   '
810   'ERROR HANDLER
820   '
830   PRINT "IBFIND ERROR"
835   END
840   PRINT "GPIB ERROR -- IBERR: ";IBERR%;"IBSTA: ";HEX$(IBSTA%)
845   END
```

EXAMPLE 3 GPIB PROGRAM FOR IBM PC (LOW-LEVEL FUNCTION CALLS)

The following example has the same function as example 2, but it is written with low level function calls.

The program assumes that the controller (board) and oscilloscope (device) are at addresses 0 and 4 respectively. The decimal listener and talker addresses of the controller and the device thus are:

	Listener address	Talker address
controller	32 (ASCII <space>)	64 (ASCII @)
device	32+4=36 (ASCII $)	64+4=68 (ASCII D).

```
1-99  <DECL.BAS>
100   CLS
110   PRINT "Control of the 9300 (address 4) via GPIB and IBM PC"
115   PRINT "": PRINT "Options :  EX to exit        LC local mode"
120   PRINT "           ST store data     RC recall data": PRINT""
125   LOOP=1
130   CMD1$ = "?_@$" 'Unlisten, Untalk, Board talker, Device listener
135   CMD2$ = "?_ D" 'Unlisten, Untalk, Board listener, Device talker
140   BDNAME$= "GPIB0": CALL IBFIND(BDNAME$,BRD0%)
145   IF BRD0% < 0 THEN GOTO 420
150   CALL IBSIC(BRD0%): IF IBSTA% < 0 THEN GOTO 425
155   WHILE LOOP
160       LINE INPUT "Enter command (EX --> Exit) : ",CMD$
165       V% = 1: CALL IBSRE(BRD0%,V%)
170       IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP = FALSE: GOTO 205
175       IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 285: GOTO 200
180       IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 365: GOTO 200
185       IF CMD$ = "lc" OR CMD$ = "LC" THEN  GOSUB 240: GOTO 200
190       IF CMD$ = "" THEN GOTO 200
195       CALL IBCMD(BRD0%,CMD1$): CALL IBWRT(BRD0%,CMD$): GOSUB 270
200   WEND
205   CALL IBSIC(BRD0%): V%=0: CALL IBSRE(BRD0%,V%)
210   CALL IBSIC(BRD0%)
215   END
220   '
230   'LOCAL MODE
235   '
240   V% = 0: CALL IBSRE(BRD0%,V%): PRINT ""
245   RETURN
250   '
260   'SUBROUTINE GET_DATA
265   '
270   CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD$): I=IBCNT%
275   FOR J=1 TO I: PRINT MID$(RD$,J,1);: NEXT J: PRINT ""
280   RETURN
285   '
```

```
290   'SUBROUTINE STORE_DATA
295   '
300   RD1$=SPACE$(3)
305   LINE INPUT "Specify trace (TA...TD,M1...M4,C1...C4):  ",TRACE$
310   LINE INPUT "Enter filename : ",FILE$
315   CALL IBCMD(BRD0%,CMD1$)
320   CMD$="WFSU NP,0,SP,0,FP,0,SN,0;CHDR SHORT"
321   CALL IBWRT(BRD0%,CMD$)
325   CMD$=TRACE$+":WF?": CALL IBWRT(BRD0%,CMD$)
330   CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD1$)
335   CALL IBRDF(BRD0%,FILE$)
340   IF IBSTA% < 0 THEN GOTO 430
345   PRINT ""
350   RETURN
355   '
360   'SUBROUTINE RECALL_DATA
365   '
370   LINE INPUT "Specify target memory (M1...M4): ",MEM$
375   LINE INPUT "Enter filename : ",FILE$
380   CALL IBCMD(BRD0%,CMD1$)
385   CMD$=MEM$+":": CALL IBWRT(BRD0%,CMD$)
390   CALL IBWRTF(BRD0%,FILE$)
395   IF IBSTA% < 0 THEN GOTO 430
400   PRINT ""
405   RETURN
410   '
415   'ERROR HANDLER
420   '
425   PRINT "IBFIND ERROR": STOP
430   PRINT "GPIB ERROR -- IBERR : ";IBERR%;"IBSTA : ";HEX$(IBSTA%)
435   STOP
440   END
```

WAVEFORM TEMPLATE

This is the response of the instrument to a command of the form TMPL?

```
TMPL "
/00
000000              LECROY_2_1:  TEMPLATE
                    8 66 109
;
; Explanation of the formats of waveforms and their descriptors on the
; LeCroy Digital Oscilloscopes,
;     Software Release 16.1.1.1, 92/03/02.
;
; A descriptor and/or a waveform consists of one or several logical data blocks
; whose formats are explained below.
; Usually, complete waveforms are read: at the minimum they consist of
;       the basic descriptor block WAVEDESC
;       a data array block.
; Some more complex waveforms, e.g. Extrema data or the results of a Fourier
; transform, may contain several data array blocks.
; When there are more blocks, they are in the following sequence:
;       the basic descriptor block WAVEDESC
;       the history text descriptor block USERTEXT (may or may not be present)
;       the time array block (for RIS and sequence acquisitions only)
;       data array block
;       auxiliary or second data array block
; With the exception of the data and time arrays, every block starts with
; an 8-character name which identifies which kind of block it is.
;
; In the following explanation, every element of a block is described by a
; single line in the form
;
; <byte position>   <variable name>: <variable type> ; <comment>
;
;  where
;
;   <byte position> = position in bytes (decimal offset) of the variable,
;                     relative to the beginning of the block.
;
;   <variable name> = name of the variable.
;
;   <variable type> = string        up to 16-character name
;                                     terminated with a null byte
;                       byte         8-bit signed data value
;                       word        16-bit signed data value
;                       long        32-bit signed data value
;                      float        32-bit IEEE floating point value
;                                   with the format shown below
;                                   31  30 .. 23   22 ... 0   bit position
;                                   s   exponent   fraction
;                                   where
;                                   s = sign of the fraction
;                                   exponent = 8 bit exponent e
;                                   fraction = 23 bit fraction f
;                                   and the final value is
;                                   (-1)**s * 2**(e-127) * 1.f
;                     double        64-bit IEEE floating point value
;                                   with the format shown below
```

```
;                                   63  62 .. 52   51 ... 0   bit position
;                                   s   exponent   fraction
;                                   where
;                                   s = sign of the fraction
;                                   exponent = 11 bit exponent e
;                                   fraction = 52 bit fraction f
;                                   and the final value is
;                                   (-1)**s * 2**(e-1023) * 1.f
;                       enum        enumerated value in the range 0 to N
;                                   represented as a 16-bit data value.
;                                   The list of values follows immediately.
;                                   The integer is preceded by an _.
;                  time_stamp       double precision floating point number,
;                                   for the number of seconds and some bytes
;                                   for minutes, hours, days, months and year.
;
;                                   double  seconds     (0 to 59)
;                                   byte    minutes     (0 to 59)
;                                   byte    hours       (0 to 23)
;                                   byte    days        (1 to 31)
;                                   byte    months      (1 to 12)
;                                   word    year        (0 to 16000)
;                                   word    unused
;                                   There are 16 bytes in a time field.
;                       data        byte, word or float, depending on the
;                                   read-out mode reflected by the WAVEDESC
;                                   variable COMM_TYPE, modifiable via the
;                                   remote command COMM_FORMAT.
;                       text        arbitrary length text string
;                                   (maximum 160)
;             unit_definition       a unit definition consists of a 48 character
;                                   ASCII string terminated with a null byte
;                                   for the unit name.
;
;=============================================================================
;
WAVEDESC: BLOCK
;
; Explanation of the wave descriptor block WAVEDESC;
;
;
<  0>          DESCRIPTOR_NAME: string  ; the first 8 chars are always WAVEDESC
;
< 16>          TEMPLATE_NAME: string
;
< 32>          COMM_TYPE: enum          ; chosen by remote command COMM_FORMAT
               _0      byte
               _1      word
               endenum
;
< 34>          COMM_ORDER: enum
               _0      HIFIRST
               _1      LOFIRST
               endenum
;
;
; The following variables of this basic wave descriptor block specify
```

```
; the block lengths of all blocks of which the entire waveform (as it is
; currently being read) is composed. If a block length is zero, this
; block is (currently) not present.
;
;
;BLOCKS :
;
< 36>          WAVE_DESCRIPTOR: long     ; length in bytes of block WAVEDESC
< 40>          USER_TEXT: long           ; length in bytes of block USERTEXT
< 44>          RES_DESC1: long           ;
;
;ARRAYS :
;
< 48>          TRIGTIME_ARRAY: long      ; length in bytes of TRIGTIME array
;
< 52>          RIS_TIME_ARRAY: long      ; length in bytes of RIS_TIME array
;
< 56>          RES_ARRAY1: long          ; an expansion entry is reserved
;
< 60>          WAVE_ARRAY_1: long        ; length in bytes of 1st simple
                                         ; data array. In transmitted waveform,
                                         ; represent the number of transmitted
                                         ; bytes in accordance with the NP
                                         ; parameter of the WFSU remote command
                                         ; and the used format (see COMM_TYPE).
;
< 64>          WAVE_ARRAY_2: long        ; length in bytes of 2nd simple
                                         ; data array
;
< 68>          RES_ARRAY2: long
< 72>          RES_ARRAY3: long          ; 2 expansion entries are reserved
;
; The following variables identify the instrument
;
< 76>          INSTRUMENT_NAME: string
;
< 92>          INSTRUMENT_NUMBER: long
;
< 96>          TRACE_LABEL: string       ; identifies the waveform.
;
<112>          RESERVED1: word
<114>          RESERVED2: word           ; 2 expansion entries
;
; The following variables describe the waveform and the time at
; which the waveform was generated.
;
<116>          WAVE_ARRAY_COUNT: long    ; number of data points in the data
                                         ; array. If there are two data
                                         ; arrays (FFT or Extrema), this number
                                         ; applies to each array separately.
;
<120>          PNTS_PER_SCREEN: long     ; nominal number of data points
                                         ; on the screen
;
<124>          FIRST_VALID_PNT: long     ; count of number of points to skip
                                         ; before first good point
                                         ; FIRST_VALID_POINT = 0
```

```
                                        ; for normal waveforms.
;
<128>           LAST_VALID_PNT: long    ; index of last good data point
                                        ; in record before padding (blanking)
                                        ; was started.
                                        ; LAST_VALID_POINT = WAVE_ARRAY_COUNT-1
                                        ; except for aborted sequence
                                        ; and rollmode acquisitions
;
<132>           FIRST_POINT: long       ; for input and output, indicates
                                        ; the offset relative to the
                                        ; beginning of the trace buffer.
                                        ; Value is the same as the FP parameter
                                        ; of the WFSU remote command.
;
<136>           SPARSING_FACTOR: long   ; for input and output, indicates
                                        ; the sparsing into the transmitted
                                        ; data block.
                                        ; Value is the same as the SP parameter
                                        ; of the WFSU remote command.
;
<140>           SEGMENT_INDEX: long     ; for input and output, indicates the
                                        ; index of the transmitted segment.
                                        ; Value is the same as the SN parameter
                                        ; of the WFSU remote command.
;
<144>           SUBARRAY_COUNT: long    ; for Sequence, acquired segment count,
                                        ; between 0 and NOM_SUBARRAY_COUNT
;
<148>           SWEEPS_PER_ACQ: long    ; for Average or Extrema,
                                        ; number of sweeps accumulated
                                        ; else 1
;
<152>           OBSOLETE1: long
;
<156>           VERTICAL_GAIN: float
;
<160>           VERTICAL_OFFSET: float  ; to get floating values from raw data :
                                        ; VERTICAL_GAIN * data - VERTICAL_OFFSET
;
<164>           MAX_VALUE: float        ; maximum allowed value. It corresponds
                                        ; to the upper edge of the grid.
;
<168>           MIN_VALUE: float        ; minimum allowed value. It corresponds
                                        ; to the lower edge of the grid.
;
<172>           NOMINAL_BITS: word      ; a measure of the intrinsic precision
                                        ; of the observation: ADC data is 8 bit
                                        ;    averaged data is 10-12 bit, etc.
;
<174>           NOM_SUBARRAY_COUNT: word ; for Sequence, nominal segment count
                                        ; else 1
;
<176>           HORIZ_INTERVAL: float   ; sampling interval for time domain
                                        ;  waveforms
;
<180>           HORIZ_OFFSET: double    ; trigger offset for the first sweep of
```

```
                                        ; the trigger, seconds between the
                                        ; trigger and the first data point
;
<188>           PIXEL_OFFSET: double    ; needed to know how to display the
                                        ; waveform
;
<196>           VERTUNIT: unit_definition  ; units of the vertical axis
;
<244>           HORUNIT: unit_definition   ; units of the horizontal axis
;
<292>           RESERVED3: word
<294>           RESERVED4: word         ; 2 expansion entries
;
<296>           TRIGGER_TIME: time_stamp ; time of the trigger
;
<312>           ACQ_DURATION: float     ; duration of the acquisition (in sec)
                                        ; in multi-trigger waveforms.
                                        ; (e.g. sequence, RIS,  or averaging)
;
<316>           RECORD_TYPE: enum
                _0      single_sweep
                _1      interleaved
                _2      histogram
                _3      trend
                _4      filter_coefficient
                _5      complex
                _6      extrema
                _7      sequence_obsolete
                endenum
;
<318>           PROCESSING_DONE: enum
                _0       no_processing
                _1       fir_filter
                _2       interpolated
                _3       sparsed
                _4       autoscaled
                _5       no_result
                _6       rolling
                _7       cumulative
                endenum
;
<320>           RESERVED5: word         ; expansion entry
;
<322>           RIS_SWEEPS: word        ; for RIS, the number of sweeps
                                        ; else 1
;
; The following variables describe the basic acquisition
; conditions used when the waveform was acquired
;
<324>           TIMEBASE: enum
                _0    1_ps/div
                _1    2_ps/div
                _2    5_ps/div
                _3    10_ps/div
                _4    20_ps/div
                _5    50_ps/div
                _6    100_ps/div
```

```
                _7    200_ps/div
                _8    500_ps/div
                _9    1_ns/div
                _10   2_ns/div
                _11   5_ns/div
                _12   10_ns/div
                _13   20_ns/div
                _14   50_ns/div
                _15   100_ns/div
                _16   200_ns/div
                _17   500_ns/div
                _18   1_us/div
                _19   2_us/div
                _20   5_us/div
                _21   10_us/div
                _22   20_us/div
                _23   50_us/div
                _24   100_us/div
                _25   200_us/div
                _26   500_us/div
                _27   1_ms/div
                _28   2_ms/div
                _29   5_ms/div
                _30   10_ms/div
                _31   20_ms/div
                _32   50_ms/div
                _33   100_ms/div
                _34   200_ms/div
                _35   500_ms/div
                _36   1_s/div
                _37   2_s/div
                _38   5_s/div
                _39   10_s/div
                _40   20_s/div
                _41   50_s/div
                _42   100_s/div
                _43   200_s/div
                _44   500_s/div
                _45   1_ks/div
                _46   2_ks/div
                _47   5_ks/div
                _100  EXTERNAL
                endenum
;
<326>           VERT_COUPLING: enum
                _0      DC_50_Ohms
                _1      ground
                _2      DC_1MOhm
                _3      ground
                _4      AC,_1MOhm
                endenum
;
<328>           PROBE_ATT: float
;
<332>           FIXED_VERT_GAIN: enum
                _0   1_uV/div
                _1   2_uV/div
```

```
                _2    5_uV/div
                _3    10_uV/div
                _4    20_uV/div
                _5    50_uV/div
                _6    100_uV/div
                _7    200_uV/div
                _8    500_uV/div
                _9    1_mV/div
                _10   2_mV/div
                _11   5_mV/div
                _12   10_mV/div
                _13   20_mV/div
                _14   50_mV/div
                _15   100_mV/div
                _16   200_mV/div
                _17   500_mV/div
                _18   1_V/div
                _19   2_V/div
                _20   5_V/div
                _21   10_V/div
                _22   20_V/div
                _23   50_V/div
                _24   100_V/div
                _25   200_V/div
                _26   500_V/div
                _27   1_kV/div
                endenum
;
<334>           BANDWIDTH_LIMIT: enum
                _0      off
                _1      on
                endenum
;
<336>           VERTICAL_VERNIER: float
;
<340>           ACQ_VERT_OFFSET: float
;
<344>           WAVE_SOURCE: enum
                _0      CHANNEL_1
                _1      CHANNEL_2
                _2      CHANNEL_3
                _3      CHANNEL_4
                _9      UNKNOWN
                endenum
;
/00               ENDBLOCK
;
;================================================================================
;
USERTEXT: BLOCK
;
; Explanation of the descriptor block USERTEXT at most 160 bytes long.
;
;
<  0>           DESCRIPTOR_NAME: string  ; the first 8 chars are always USERTEXT
;
< 16>           TEXT: text               ; a list of ASCII characters
```

```
;
/00              ENDBLOCK
;
;=============================================================================
;
DATA_ARRAY_1: ARRAY
;
; Explanation of the data array DATA_ARRAY_1.
; The data item is repeated for each acquired or computed data point
; of the first data array of any waveform.
;
<  0>           MEASUREMENT: data          ; the actual format of a data is
                                           ; given in the WAVEDESC descriptor
                                           ; by the COMM_TYPE variable.
;
/00               ENDARRAY
;
;=============================================================================
;
DATA_ARRAY_2: ARRAY
;
; Explanation of the data array DATA_ARRAY_2.
; The data item is repeated for each acquired or computed data point
; of the second data array of any dual waveform
; e.g. the imaginary part of a FFT.
;
<  0>           MEASUREMENT: data          ; the actual format of a data is
                                           ; given in the WAVEDESC descriptor
                                           ; by the COMM_TYPE variable.
;
/00               ENDARRAY
;
;=============================================================================
;
TRIGTIME: ARRAY
;
; Explanation of the trigger time array TRIGTIME.
; This data block is repeated for each segment which makes up the acquired
; sequence record.
;
<  0>           TRIGGER_TIME: double       ; for sequence acquisitions,
                                           ; time in seconds from first
                                           ; trigger to this one
;
<  8>           TRIGGER_OFFSET: double     ; the trigger offset is in seconds
                                           ; from trigger to zeroth data point
;
/00               ENDARRAY
;
;=============================================================================
;
RISTIME: ARRAY
;
; Explanation of the random-interleaved-sampling (RIS) time array RISTIME.
; This data block is repeated for each sweep which makes up the RIS record
;
<  0>           RIS_OFFSET: double         ; seconds from trigger to zeroth
```

```
                                        ; point of segment
;
/00             ENDARRAY
;
;=============================================================================
;
SIMPLE: ARRAY
;
; Explanation of the data array SIMPLE.
; The data item is repeated for each acquired or computed data point
; of the first data array of any waveform.
; It is identical to DATA_ARRAY_1 above.
;
<  0>          MEASUREMENT: data       ; the actual format of a data is
                                        ; given in the WAVEDESC descriptor
                                        ; by the COMM_TYPE variable.
;
/00              ENDARRAY
;
;=============================================================================
;
DUAL: ARRAY
;
; Explanation of the DUAL array.
; This data block is repeated for each pair of computed data points
; composing respectively the first and second data array of a waveform,
; e.g the real and imaginary parts of an FFT.
;
<  0>          MEASUREMENT_1: data     ; data in the first data array.
                                        ; the actual format of a data is
                                        ; given in the WAVEDESC descriptor
                                        ; by the COMM_TYPE variable.
;
<  0>          MEASUREMENT_2: data     ; data in the second data array
                                        ; the byte offset depends on
                                        ; the actual format of the data which
                                        ; is given in the WAVEDESC descriptor
                                        ; by the COMM_TYPE variable.
;
/00              ENDARRAY
;
;
000000               ENDTEMPLATE
"
```

A

ALL_STATUS?, ALST, Query, 5–5
ARM_ACQUISITION, ARM, Command, 5–6
Assistance, 1–1
ATTENUATION, ATTN, Command/Query, 5–7
AUTO_CALIBRATE, ACAL, Command/Query, 5–8
AUTO_SETUP, ASET, Command, 5–9

B

BANDWIDTH_LIMIT, BWL, Command/Query, 5–10
BASIC(A), 3–5, 6–4, A–2
Binary Blocks, 6–5
BUZZER, BUZZ, Command, 5–11

C

CAL_OUTPUT, COUT, Command/Query, 5–13
*CAL? Query, 5–12
CALL_HOST, CHST, Command/Query, 5–14
CLEAR_SWEEPS, CLSW, Command, 5–15
*CLS Command, 5–16
CMR – CoMmand error Register, 7–1, 7–3, 7–4, 7–6
CMR? Query, 5–17
COMM_FORMAT, CFMT, Command/Query, 5–18
COMM_HEADER, CHDR, Command/Query, 5–20
COMM_HELP, CHLP, Command/Query, 5–21
COMM_ORDER, CORD, Command/Query, 5–22
COMM_RS232, CORS, Command/Query, 5–23
CoMmand error Register, 7–6
Command Execution, 5–3
Command Notation, 5–4
Command Summary, 5–1
Commands
  ALL_STATUS?, ALST, 5–5
  ARM_ACQUISITION, ARM, 5–6
  ATTENUATION, ATTN, 5–7
  AUTO_CALIBRATE, ACAL, 5–8
  AUTO_SETUP, ASET, 5–9
  BANDWIDTH_LIMIT, BWL, 5–10
  BUZZER, BUZZ, 5–11
  *CAL?, 5–12
  CAL_OUTPUT, COUT, 5–13
  CALL_HOST, CHST, 5–14
  CLEAR_SWEEPS, CLSW, 5–15
  *CLS, 5–16
  CMR?, 5–17
  COMM_FORMAT, CFMT, 5–18
  COMM_HEADER, CHDR, 5–20
  COMM_HELP, CHLP, 5–21
  COMM_ORDER, CORD, 5–22
  COMM_RS232, CORS, 5–23
  COUPLING, CPL, 5–26
  CURSOR_MEASURE, CRMS, 5–27
  CURSOR_SET, CRST, 5–29
  CURSOR_VALUE?, CRVA?, 5–31
  DATE, 5–32
  DDR?, 5–33
  DEFINE, DEF (Oscilloscopes fitted with WP01), 5–36
  DEFINE, DEF (Oscilloscopes fitted with WP02), 5–39
  DEFINE, DEF (Standard Oscilloscopes), 5–34
  DELETE_FILE, DELF, 5–41
  DIRECTORY_LIST?, DIR?, 5–42
  DISPLAY, DISP, 5–43
  DUAL_ZOOM, DZOM, 5–44

*ESE, 5–45
*ESR?, 5–46
EXR?, 5–48
FORMAT_CARD, FCRD, 5–50
FORMAT_FLOPPY, FFLP, 5–52
FUNCTION_RESET, FRST, 5–54
GRID, 5–55
HARDCOPY_SETUP, HCSU, 5–56
HARDCOPY_TRANSMIT, HCTR, 5–58
HOR_MAGNIFY, HMAG, 5–59
HOR_POSITION, HPOS, 5–60
*IDN?, 5–62
INE, 5–63
INR?, 5–64
INSPECT?, INSP?, 5–65
INTENSITY, INTS, 5–67
INTERLEAVED, ILVD, 5–68
*IST?, 5–69
KEY, 5–70
MEMORY_SIZE, MSIZ, 5–72
MESSAGE, MSG, 5–73
MULTI_ZOOM, MZOM, 5–74
OFFSET, OFST, Command/Query, 5–75
*OPC, 5–76
*OPT?, 5–77
PANEL_SETUP, PNSU, 5–78
PARAMETER_CLR, PACL, 5–79
PARAMETER_DELETE, PADL, 5–80
PARAMETER_STATISTICS?, PAST?, 5–81
PARAMETER_VALUE?, PAVA?, 5–82
PASS_FAIL_CONDITION, PFCO, 5–84
PASS_FAIL_COUNTER, PFCT, 5–86
PASS_FAIL_DO, PFDO, 5–87
PASS_FAIL_MASK, PFMS, 5–89
PER_CURSOR_SET, PECS, 5–91
PER_CURSOR_VALUE?, PECV?, 5–93
PERSIST, PERS, 5–90
*PRE, 5–94
*RCL, 5–95
RECALL, REC, 5–96
RECALL_PANEL, RCPN, 5–97
*RST, 5–98
SAMPLE_CLOCK, SCLK, 5–99
*SAV, 5–100
SCREEN_DUMP, SCDP, 5–101
SEGMENTS, SEGS, 5–102
SELECT, SEL, 5–103
SEQUENCE, SEQ, 5–104
*SRE, 5–105
*STB?, 5–106
STOP, 5–108
STORE, STO, 5–109
STORE_PANEL, STPN, 5–110
STORE_SETUP, STST, 5–111
STORE_TEMPLATE, STTM, 5–112
TEMPLATE?, TMPL?, 5–113
TIME_DIV, TDIV, 5–114
TRACE, TRA, 5–116
*TRG, 5–117
TRIG_COUPLING, TRCP, 5–118
TRIG_DELAY, TRDL, 5–119
TRIG_LEVEL, TRLV, 5–120
TRIG_MODE, TRMD, 5–121
TRIG_SELECT, TRSE, 5–122
TRIG_SLOPE, TRSL, 5–125
TRIG_WINDOW, TRWI, 5–126
*TST?, 5–127
URR?, 5–128
VERT_MAGNIFY, VMAG, 5–129
VERT_POSITION, VPOS, 5–130
VOLT_DIV, VDIV, 5–131
*WAI, 5–132
WAIT, 5–133
WAVEFORM, WF, 5–134
WAVEFORM_SETUP, WFSU, 5–136
WAVEFORM_TEXT, WFTX, 5–138
XY_ASSIGN?, XYAS?, 5–139
XY_CURSOR_ORIGIN, XYCO, 5–140
XY_CURSOR_SET, XYCS, 5–141
XY_CURSOR_VALUE?, XYCV?, 5–143
XY_DISPLAY, XYDS, 5–144
Commands and Queries, 2–2, 2–4
Continuous Polling, 3–11
Controller Timeout, 2–1, 3–3, 3–9, 3–12, A–2

COUPLING, CPL, Command/Query, 5–26
CURSOR_MEASURE, CRMS, Command/Query, 5–27
CURSOR_SET, CRST, Command/Query, 5–29
CURSOR_VALUE?, CRVA?, Query, 5–31
Customer Service, 1–1

D

Data
- Arrays, 6–1, 6–2
- ASCII forms, 2–5
- Blocks, 6–1
- Formatting, 6–4, 6–5, 6–10
- HEX mode, 4–1, 4–5, 6–4, 6–10
- Horizontal position, 6–7
- Interpretation, 6–4, 6–6
- Sparsing, 6–10
- Time of, 6–7
- Values, 6–3, 6–5

DATE Command/Query, 5–32
DDR – Device Dependent error status Register, 7–6
DDR? Query, 5–33
DEFINE, DEF, Command/Query (Oscilloscopes fitted with WP01), 5–36
DEFINE, DEF, Command/Query (Oscilloscopes fitted with WP02), 5–39
DEFINE, DEF, Command/Query (Standard Oscilloscopes), 5–34
DELETE_FILE, DELF, Command, 5–41
Descriptor
- Block, 6–1, 6–5
- Values, 6–3, 6–5

Device Dependent error status Register, 7–6
Diagnostics Help Messages, 2–2
DIRECTORY_LIST?, DIR?, Query, 5–42
DISPLAY, DISP, Command/Query, 5–43
DUAL, 6–2, 6–3
DUAL_ZOOM, DZOM, Command/Query, 5–44

E

Error Messages, 2–1
*ESE Command/Query, 5–45
ESE – Standard Event Status Enable register, 3–10, 7–1, 7–3, 7–5
*ESR? Query, 5–46
ESR – Standard Event Status Register, 3–10, 7–1, 7–4
EXecution error status Register, 7–6
EXR – EXecution error Register, 7–1
EXR – EXecution error status Register, 7–6
EXR? Query, 5–48

F

FORMAT_CARD, FCRD, Command/Query, 5–50
FORMAT_FLOPPY, FFLP, Command/Query, 5–52
FUNCTION_RESET, FRST, Command, 5–54

G

GPIB, 3–1
- Addresses, 3–2
- ATN – ATteNtion, 3–2
- Data lines, 3–2
- DCL – Device CLear, 3–4
- EOI – End Or Identify, 2–3, 3–3
- GET – Group Execute Trigger, 3–4, 3–9
- GTL – Go To Local, 3–4, 3–8
- Handshake lines, 3–2
- Hard copies, 3–15
- Hardware Configuration, 3–5

IEEE 488.1, 2–1, 3–3, 3–14
IEEE 488.2, 2–1, 3–3, 7–1, 7–4
IFC – InterFace Clear, 3–2, 3–5
Interface capabilities, 3–1
Interface management lines, 3–2
Listener address, 3–14
LLO – Local LOckout, 3–4
MLA – Listen address, 3–2, 3–14, 3–17
MTA – Talker address, 3–2, 3–14, 3–17
Polling, 3–11
Programs for GPIB, 3–6, 3–11, A–1
REN – Remote ENable, 3–2, 3–4
RQS – ReQuest for Service, 3–12
SDC – Selected Device Clear, 3–4, 3–8
Signals, 3–2
Software Configuration, 3–6
SRE – Service Request Enable register, 3–9, 3–12
SRQ – Service ReQuest, 3–3, 3–9
Talker address, 3–14
Transfers, 3–5
UNL – Universal unlisten, 3–2, 3–14, 3–17
UNT – Universal untalk, 3–2, 3–14, 3–17
GRID Command/Query, 5–55

H

Hard Copies. *See* GPIB
HARDCOPY_SETUP, HCSU, Command/Query, 5–56
HARDCOPY_TRANSMIT, HCTR, Command, 5–58
Header, 2–4
Header Path, 2–4
Help Messages, 2–2
HOR_MAGNIFY, HMAG, Command/Query, 5–59
HOR_POSITION, HPOS, Command/Query, 5–60

I

IBCLR, 3–8
IBCMD, 3–13
IBFIND, 3–7
IBLOC, 3–8
IBRD, 3–7
IBRDF, 3–8
IBRDI, 3–9
IBRPP, 3–13
IBRSP, 3–12
IBTMO, 3–9
IBTRG, 3–9
IBWAIT, 3–9, 3–12
IBWRT, 3–7
IBWRTF, 3–8
IBWRTI, 3–9
*IDN? Query, 5–62
IEEE Standards. *See* GPIB
INE – INternal state change Enable register, 3–10, 3–12, 3–13, 3–14, 7–1, 7–3, 7–6
INE Command/Query, 5–63
INR – INternal state change Register, 3–10, 3–11, 3–12, 3–13, 3–14, 7–1, 7–3, 7–5
INR? Query, 5–64
INSPECT? Queries, 6–2
INSPECT?, INSP?, Query, 5–65
INTENSITY, INTS, Command/Query, 5–67
Interface Messages, 3–1
INTERLEAVED, ILVD, Command/Query, 5–68
INternal state change Enable register, 7–6
INternal state change Register, 7–5
IST Polling, 3–14, 7–3, 7–5
*IST? Query, 5–69

K

KEY Command, 5–70

L

Line Splitting. *See* RS–232–C
Local State, 2–2
Logical Data Blocks, 6–1

M

Maintenance, 1–1
MEMORY_SIZE, MSIZ, Command/Query, 5–72
MESSAGE, MSG, Command/Query, 5–73
MULTI_ZOOM, MZOM, Command/Query, 5–74

O

OFFSET, OFST, Command/Query, 5–75
*OPC Command/Query, 5–76
*OPT? Query, 5–77

P

PANEL_SETUP, PNSU, Command/Query, 5–78
PaRallel poll Enable register, 7–5
Parallel Polling, 3–13—3–14
Parameter measurements, 5–28
PARAMETER_CLR, PACL, Command, 5–79
PARAMETER_DELETE, PADL, Command, 5–80
PARAMETER_STATISTICS?, PAST?, Query, 5–81
PARAMETER_VALUE?, PAVA?, Query, 5–82
PASS_FAIL_CONDITION, PFCO, Command/Query, 5–84
PASS_FAIL_COUNTER, PFCT, Command/Query, 5–86
PASS_FAIL_DO, PFDO, Command/Query, 5–87
PASS_FAIL_MASK, PFMS, Command/Query, 5–89
PER_CURSOR_SET, PECS, Command/Query, 5–91
PER_CURSOR_VALUE?, PECV?, Query, 5–93
PERSIST, PERS, Command/Query, 5–90
Pin Assignments. *See* RS–232–C
Polling, 3–11
*PRE Command/Query, 5–94
PRE – PaRallel poll Enable register, 3–13, 3–14, 7–3, 7–5
Program Messages, 2–1, 2–3

R

RAN – Return Authorization Number, 1–2
*RCL Command, 5–95
RECALL, REC, Command, 5–96
RECALL_PANEL, RCPN, Command, 5–97
Remote State, 2–2
Reset Instrument, 5–4
Response Message Form, 2–7
Return Procedure, 1–2
RISTIME, 6–2, 6–9
RS–232–C, 4–1
  Configuration, 4–2
  Echoing, 4–2
  Editing, 4–3
  Handshake control, 4–2
  Immediate commands, 4–2
  Line splitting, 4–4
  Message terminators, 4–3
  Pin assignments, 4–1
  Simulating GPIB commands, 4–5
  SRQ – Service request, 4–4
*RST Command, 5–98

S

SAMPLE_CLOCK, SCLK, Command/Query, 5–99
*SAV Command, 5–100
SCREEN_DUMP, SCDP, Command/Query, 5–101
SEGMENTS, SEGS, Command/Query, 5–102
SELECT, SEL, Command/Query, 5–103
SEQUENCE, SEQ, Command/Query, 5–104
Serial Polling, 3–12
Service Procedure, 1–2
Service Request Enable register, 7–5
Service Requests, 7–1
SIMPLE, 6–2
*SRE Command/Query, 5–105
SRE – Service Request Enable register, 3–10, 7–1, 7–3, 7–5
SRQ – Service ReQuest, 3–9, 4–4, 7–3
Standard Event Status Enable Register. *See* ESE
Standard Event Status Enable register, 7–5
Standard Event Status Register, 7–4
See also ESR
STatus Byte register, 7–3
Status Register Reporting, 7–1
*STB? Query, 5–106
STB – STatus Byte register, 3–10, 4–4, 7–3
STOP Command, 5–108
STORE, STO, Command, 5–109
STORE_PANEL, STPN, Command, 5–110
STORE_SETUP, STST, Command/Query, 5–111
STORE_TEMPLATE, STTM, Command, 5–112
Suffix Multipliers, 2–6

T

Template, 6–1, 6–2, 6–5, 6–6, 6–7, B–1—B–9
TEMPLATE?, TMPL?, Query, 5–113
Terminators, 2–3, 2–7, 4–3, 6–6
TIME_DIV, TDIV, Command/Query, 5–114
TRACE, TRA, Command/Query, 5–116
Transfers, 3–5
*TRG Command, 5–117
TRIG_COUPLING, TRCP, Command/Query, 5–118
TRIG_DELAY, TRDL, Command/Query, 5–119
TRIG_LEVEL, TRLV, Command/Query, 5–120
TRIG_MODE, TRMD, Command/Query, 5–121
TRIG_SELECT, TRSE, Command/Query, 5–122
TRIG_SLOPE, TRSL, Command/Query, 5–125
TRIG_WINDOW, TRWI, Command/Query, 5–126
Trigger Time, 6–2
TRIGTIME, 6–1, 6–8
*TST? Query, 5–127

U

URR – User Request status Register, 3–10, 7–6
URR? Query, 5–128
User Request status Register, 7–6
USERTEXT, 6–1

V

VERT_MAGNIFY, VMAG, Command/Query, 5–129
VERT_POSITION, VPOS, Command/Query, 5–130
VOLT_DIV, VDIV, Command/Query, 5–131

W

*WAI Command, 5–132
WAIT Command, 5–133
Warning Messages, 2–1
Warranty, 1–1
WAVEDESC. *See* Descriptor
WAVEFORM
  Command, 6–9
  Query, 6–4, 6–10
  Transfer Optimization, 6–10
Waveform Template, B–1—B–9
WAVEFORM, WF, Command/Query, 5–134
WAVEFORM_SETUP, WFSU, Command/Query, 5–136
WAVEFORM_TEXT, WFTX, Command/Query, 5–138

X

XY_ASSIGN?, XYAS?, Query, 5–139
XY_CURSOR_ORIGIN, XYCO, Command/Query, 5–140
XY_CURSOR_SET, XYCS, Command/Query, 5–141
XY_CURSOR_VALUE?, XYCV?, Query, 5–143
XY_DISPLAY, XYDS, Command/Query, 5–144

US SALES OFFICES

1 800–5–LeCroy (automatically connects you to your local sales office)

WORLDWIDE

Argentina: Search SA, (01) 46 6156

Australia: Scientific Devices Pty.Ltd, (03) 579–3622

Austria: Dewetron GmbH, (0316) 391804

Benelux: LeCroy B.V., 4902 8 9285

Brazil: ATP–Hi–Tek, (011) 725.5822

Canada: LeCroy Canada Inc, (514) 928–4707

Chile: Sistemas de Instrumentacion (562) 695.1137

Denmark: Lutronic, (42) 459764

Eastern Europe: Elsinco GmbH, Vienna 222 812 1751

Finland: Orbis OY, (90) 566.4066

France: LeCroy Sarl (16)907 3897

Germany: LeCroy GmbH, (06221) 831001 (North) (0405) 42713

Greece: Hellenic S/R Ltd., (01) 721 1140

Hong Kong: Euro Tech (Far East) Ltd. 8.52.814.0311

India: TATA–Honeywell, (212) 670.445

Indonesia: P.T. DWI, (6221) 570.4563

Israel: Ammo, (03) 453157

Italy: LeCroy S.r.l.,Roma (06) 336.797.00
Milano (02) 204 7082
(02) 2940–5634

Japan: LeCroy Japan,
Tokyo (0081) 3 376 9400
Osaka (0081) 6 330 0961

Korea: Woojoo Hi–Tech Corp. (02) 82.2.449.5472

Mexico: Nucleoelectronica SA: (905) 593 6043

New Zealand: E.C. Gough Ltd., (03) 379.8740

Norway: Avantec AS, (02) 630520

Pakistan: Electro Tech Corp.Ltd., (021) 493 8087

Portugal: M.T. Brandao, Lda., (02) 815680

Singapore: Singapore Elec. & Eng. Ltd., (065) 283.6288

S. Africa: Westplex, (011)787 0473

Spain: Essa, (01)359 0088

Sweden: MSS AB, (0764) 68100

Switzerland: LeCroy S.A., (022) 719 2111

Taiwan: Topward El. Inst., Ltd., (02) 601 8801

Thailand: Measuretronix Ltd., (02) 3752733–4

United Kingdom: LeCroy Ltd., (23) 553 3114

LeCROY CORPORATE HEADQUARTERS

700 Chestnut Ridge Road
Chestnut Ridge, NY 10977–6499
Telephone: (914) 425–2000
TWX:(710) 577–2832
Fax: (914) 425–8967

LeCROY EUROPEAN HEADQUARTERS

2, rue du Pré–de–la–Fontaine
P.O. Box 341
1217 Meyrin 1/Geneva, Switzerland
Tel.: (022) 719 21 11 Telex: 419 058
Fax: (022) 782 39 15

LeCroy

www.manualspius.com

1004264